Théorème de Fagin

Cyril BOUVIER

18 décembre 2008

Table des matières

Introduction

Le théorème de Fagin est un théorème qui énonce une équivalence entre logique et complexité. Le théorème affirme que les problèmes **NP** sont exactement ceux que l'on peut exprimer en logique existentielle du second ordre. Nous définirons plus loin ce que ceci signifie formellement.

Les machines de Turing considérées dans la suite seront des machines à une bande dont le symbole blanc est représenté par $\sharp$, dont l'état initial est s et qui accepte uniquement sur l'état q_+ et rejette uniquement sur l'état q_-. Pour les machines de Turing non déterministes on supposera que pour chaque transition il y a deux choix non déterministes possibles. On peut supposer tout ceci sans perte de généralité, car toute machine de Turing est équivalente à une machine avec les hypothèses ci-dessus.

1 Correspondance entre logique et problèmes de graphes

1.1 Logique du première ordre et logique existentielle du second ordre ($\exists$SO)

Définition 1.1. Logique du première ordre

- Un langage Σ est un ensemble contenant des symboles de fonctions k-aire, $k \in \mathbb{N}$ (les fonctions 0-aire correspondent aux constantes), des symboles de relations k-aire, $k \in \mathbb{N}^*$ et un ensemble dénombrable de variables $(v_i)_{i\in\mathbb{N}}$. On suppose de plus que l'on dispose toujours d'une relation 2-aire notée $=$.

- On définit les termes de la logique du première ordre par induction. Les variables v_i pour $i \in \mathbb{N}$ sont des termes, et pour $k \in \mathbb{N}$ si f est un symbole de fonction k-aire et $t_1, \ldots, t_k$ sont des termes $f(t_1, \ldots, t_k)$ est un terme.

- Pour $k \in \mathbb{N}^*$ si R est un symbole de relation k-aire et $t_1, \ldots, t_k$ sont des termes, alors $R(t_1, \ldots, t_k)$ est appelé formule atomique.

- On définit enfin les formules de la logique du première ordre par induction. C'est le plus petit ensemble contenant les formules atomiques et stable par $\neg$ (non logique), $\wedge$ (et logique) , $\vee$ (ou logique) , $\Rightarrow$, $\Leftrightarrow$, $\forall v_i$ et $\exists v_i$.

Remarque 1.2. Pour la définition des formules, on pourrait considérer uniquement la négation, le "et" et le $\forall$, car les autres s'en déduisent. Pour faciliter l'écriture des formules on considère tous les connecteurs logiques, mais pour les preuves par induction sur les formules, on se contentera de vérifier la stabilité par $\neg$, $\wedge$ et $\forall v_i$

Pour interpréter une formule, on se donne un modèle $\mathcal{M}$ qui contient un ensemble M et les interprétations des symboles de fonctions et de relations : à f un symbole de fonction k-aire, on associe une fonction de M^k dans M notée f^M et à R un symbole de relation k-aire, on associe une partie de M^k notée R^M. On suppose que la relation $=$ est bien interprétée comme l'égalité sur M.

Si dans le modèle $\mathcal{M}$, une formule ϕ est vraie on note $\mathcal{M} \models \phi$.

Exemple 1.3. On prend le langage ne comportant qu'une relation binaire G, c'est le langage de la théorie des graphes, noté Σ_G. Un modèle $\mathcal{M}$ de cette théorie peut se voir comme un graphe, l'ensemble M sous-jacent au modèle est l'ensemble des sommets et l'interprétation de G est la relation d'adjacence des sommets.

La formule $\Phi = \forall x \forall y (G(x, y) \Rightarrow G(y, x))$ sera vérifiée pour les modèles représentés par des graphes symétriques.

Définition 1.4. ($\exists$SO)
Les formules de la logique existentielle du second ordre sont de la forme $\exists P \phi$ où
- P est un nouveau symbole de relation qui n'est pas contenu dans Σ.
- ϕ est une formule du première ordre sur le langage $\Sigma' = \Sigma \cup P$.

L'interprétation d'une formule $\exists P\phi$ dans une structure $\mathcal{M}$ se fait de la façon suivant : $\mathcal{M} \models \exists P\phi$ si et seulement s'il existe une relation P^M tel que $\mathcal{M}' \models \phi$ où $\mathcal{M}' = \mathcal{M} \cup P^M$.

Exemple 1.5. Soit $\gamma = \exists P \forall x \forall y((P(x,y) \Rightarrow G(x,y)) \wedge (P(x,y) \Rightarrow P(y,x)))$. Un modèle $\mathcal{M}$ tel que $\mathcal{M} \models \gamma$ est représenté par un graphe admettant un sous graphe symétrique.

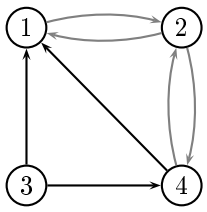

FIG. 1 – Graphe représentant un modèle $\mathcal{M}$ tel que $\mathcal{M} \models \gamma$ (le sous graphe représentant P est en gris)

1.2 Problèmes ϕ-GRAPHS

On va maintenant faire le lien entre la logique et les problèmes de graphes en associant à chaque formule logique un problème de graphes.

Définition 1.6. (Problèmes ϕ-GRAPHS)

Pour ϕ une formule de la logique du première ordre sur le langage Σ_G, on associe le problème ϕ-GRAPHS suivant : étant donné un modèle Γ, la formule ϕ est-elle vérifiée dans Γ, c'est à dire a-t-on $\Gamma \models \phi$?

Pour un formule de $\exists$SO de la forme $\phi = \exists P\psi$ on associe le problème ϕ-GRAPHS suivant : étant donné un modèle Γ, existe-t-il une relation qui interprète P et tel que l'on ait $\Gamma \cup P^\Gamma \models \psi$?

Remarque 1.7. On définit les problèmes ϕ-GRAPHS pour des formules qui peuvent contenir des variables libres, il faut donc dans le modèle rajouter des informations associant à chaque variable libre de la formule un sommet du graphe. On définit les problèmes ϕ-GRAPHS ainsi pour pouvoir effectuer des preuves par induction sur les formules (cf preuve du lemme 2.3). Par exemple pour la formule $\alpha(y) = \forall x \neg(x = y) \Rightarrow G(x,y)$ il faut associer à y un sommet du graphe dans le modèle Γ. Pour un même graphe, l'association de y à deux sommets différents correspond à deux modèles différents donc potentiellement à deux réponses différentes du problème ϕ-GRAPHS associé.

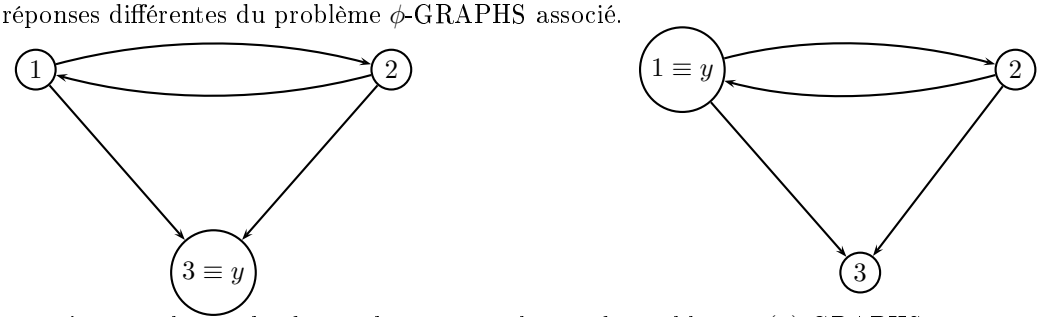

FIG. 2 – Il n'y a que le graphe de gauche qui est solution du problème $\alpha(y)$-GRAPHS.

Exemple 1.8. Reprenons la formule Φ de l'exemple 1.3 : $\Phi = \forall x \forall y (G(x,y) \Rightarrow G(y,x))$. Le problème Φ-GRAPHS associé est le fait de déterminer pour un graphe donné en entrée s'il est symétrique.

Considérons maintenant la formule γ de l'exemple 1.5 : $\gamma = \exists P \forall x \forall y ((P(x,y) \Rightarrow G(x,y)) \wedge (P(x,y) \Rightarrow P(y,x)))$. Le problème γ-GRAPHS associé est le fait de déterminer pour un graphe donné en entrée s'il existe un sous graphe symétrique.

Remarque 1.9. Justifions maintenant l'introduction de la logique existentielle du second ordre. Considérons le problème suivant : existe-t-il une formule $\phi(x,y)$ tel que le problème $\phi(x,y)$-GRAPHS associé soit de déterminer s'il existe un chemin de x vers y dans le graphe-modèle G (x et y sont des variables libres de ϕ et doivent donc avoir une correspondance avec des sommets du graphe via le modèle). On peut montrer grâce au théorème de Löwenheim-Skolem (qui est vrai dans le cas de la logique du première ordre mais pas dans le cas de la logique existentielle du second ordre) que $\phi(x,y)$ ne peut être une formule du première ordre. On voit donc la nécessité d'une logique ayant un pouvoir d'expression plus fort. La logique existentielle du second ordre permet d'exprimer cette propriété et même plus (comme l'existence d'un cycle Hamiltonien) car elle permet de quantifier sur autre chose que sur les variables-sommets v_i.

Dans la partie suivante, nous énoncerons et démontrerons le théorème de Fagin reliant la logique existentielle du second ordre et la classe **NP** .

2 Le théorème de Fagin

On définit une propriété sur les graphes comme un ensemble $\mathcal{G}$ de graphes. Le problème correspondant est de déterminer pour un graphe G donné si $G \in \mathcal{G}$. On dit que $\mathcal{G}$ est exprimable en logique existentielle du second ordre s'il existe ϕ dans $\exists$SO tel que $G \models \phi$ si et seulement si $G \in \mathcal{G}$ (ie si le problème associé à $\mathcal{G}$ est le problème ϕ-GRAPHS).

Théorème 2.1 (Fagin (1974))**.** *La classe de toutes les propriétés sur les graphes exprimables en logique existentielle du second ordre est précisément la classe* ***NP*** .

Remarque 2.2. Tous les problèmes calculables ne sont pas des problèmes sur les graphes (en particulier **P** et **NP** ne contiennent pas que des problèmes sur les graphes) mais cela résulte du fait de notre choix d'encodage des problèmes. On aurait tout aussi bien pu décider de coder nos problèmes par des graphes. On peut donc supposer ici que **P** et **NP** contiennent uniquement des problèmes de graphes. Ou l'on peut reformuler le théorème s'il on ne veut pas adopter ce nouveau point de vu. Le théorème deviendrait donc :

La classe **NP** est précisément la classe de tous les problèmes réductibles à un problème de graphes exprimable en logique existentielle du second ordre.

2.1 Cas de la logique du première ordre et début de la preuve du théorème

Lemme 2.3. *Soit ϕ une formule du première ordre, alors le problème ϕ-GRAPHS correspondant est dans* ***P*** .

Démonstration. Par induction sur ϕ

- Si ϕ est atomique, ie $\phi = G(x,x)$ ou $\phi = G(x,y)$ alors le résultat est évident.
- Si $\phi = \neg\psi$, il suffit de renvoyer l'inverse de ce que renvoie la machine de Turing qui calcule ψ-GRAPHS (c'est possible car elle s'arrête sur chaque entrée). On ne change pas la complexité.
- Si $\phi = \psi_1 \wedge \psi_2$, on lance la machine de Turing qui calcule ψ_1-GRAPHS, puis celle qui calcule ψ_2-GRAPHS et on renvoie "vrai" si les deux ont renvoyé "vrai", et "faux" sinon. On reste bien dans **P**
- Si $\phi = \forall x\psi$, on sait qu'il existe une machine de Turing qui résout le problème ψ-GRAPHS en temps polynomial. On remarque aussi que ψ contient x comme variable libre. On définit l'algorithme suivant pour ϕ-GRAPHS : pour chaque sommet v du graphe G on fait correspondre x à v dans le modèle Γ et on teste ensuite avec l'algorithme pour ψ si ce nouveau modèle satisfait le problème ψ-GRAPHS. On répond "vrai" si l'algorithme pour ψ répond "vrai" pour tous les associations de x à v, sinon on répond "faux". Si l'algorithme pour ψ est en $\mathcal{O}(n^d)$ (n est le nombre de sommets), l'algorithme pour ϕ est en $\mathcal{O}(n^{d+1})$, on reste bien dans **P** .

□

On peut maintenant montrer un sens du théorème de Fagin.

Corollaire 2.4. *Si $\phi = \exists P\psi$, alors le problème ϕ-GRAPHS correspondant est dans **NP** , ie la classe de toutes les propriétés sur les graphes exprimables en logique existentielle du second ordre est inclus dans la classe **NP** .*

Démonstration. On appelle k l'arité de P (ie P est une relation k-aire). On construit une machine de Turing non déterministe qui prend un modèle Γ en paramètre (un graphe G à n sommets) et qui renvoie si $\Gamma \models \exists P\psi$: la machine devine un relation P^Γ tel que ψ est satisfaite dans le modèle Γ auquel on ajoute P^Γ. D'après le lemme précédent la vérification se fait en temps polynomial, et de plus le fait de deviner P^Γ se fait aussi polynomialement en n car il faut deviner au plus n^k éléments. □

On a donc montrer un sens du théorème de Fagin, avant d'étudier la réciproque il faut introduire une nouvelle notion utile pour la preuve.

2.2 Table de configurations

On va maintenant définit une notion que l'on utilisera pour la preuve de la réciproque.

Définition 2.5. Soit M une machine de Turing (déterministe ou non) qui décide un langage en temps polynomial $\mathcal{O}(n^k)$. Quitte à modifier M on peut supposer que tout calcul prend exactement un temps n^k pour une entrée de longueur n. Ainsi la bande contient au maximum n^k caractères. On construit une matrice $n^k \times n^k$ où la $i$ème ligne représente l'état de la bande au temps i, que l'on complète par des $\sharp$ pour avoir exactement n^k caractères. De plus pour tout temps $i \in [0, n^k - 1]$ si la machine lit le $j$ème caractère a et est dans l'état q, on note le (i,j)ème caractère a_q.

Exemple 2.6. On considère la machine M sur l'alphabet de bande $\{0,1\}$ qui accepte $\{0,1\}^*$. Pour que tous les calculs se fassent exactement en un temps n^1, la machine M part de l'état initial s, arrive à l'état final acceptant q_+ en parcourant toute la bande. On

considère l'entrée suivante : 00110010. On aura donc la table de configurations de taille 8×8 suivante :

$$\begin{array}{cccccccc}
0_s & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0_s & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1_s & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1_s & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0_s & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0_s & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1_s & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0_{q_+}
\end{array}$$

2.3 Fin de la preuve

Démontrons maintenant la réciproque.

Proposition 2.7. *Tous problèmes de la classe **NP** est un problème ϕ-GRAPHS pour un certain ϕ dans $\exists SO$.*

Démonstration. [1] On se donne $\mathcal{G}$ un problème sur les graphes dans **NP** , on cherche à construire un $\phi = \exists P\psi$ dans $\exists$SO tel que pour tout graphe G, $G \in \mathcal{G}$ si et seulement si $G \models \exists P\psi$. Soit M une machine de Turing non déterministe qui décide $\mathcal{G}$: elle prend en entrée G un graphe à n sommets et décide si $G \in \mathcal{G}$ en $\mathcal{O}(n^k)$ pour un certain k. On suppose, quitte à modifier la machine M, que $k > 2$ et que pour toute entrée de taille m la machine calcule en un temps m^k (la dernière hypothèse permet l'utilisation des tables de configurations définies précédemment).

On fait encore quelques hypothèses sur la machine M sans perte de généralité. On suppose que l'entrée est la suite des lignes de la matrice d'adjacence du graphe G où chaque entrée est suivie par $n^{k-2} - 1$ caractères $\sharp$, ainsi le graphe est représenté sur la bande d'entrée par n^k caractères. On va ensuite chercher la formule ϕ sous la forme $\exists P_1 \exists P_2 ... \exists P_r \psi$, la relation P cherchée sera le produit cartésien des relations P_i

Tout d'abord on définit un nouveau symbole de relation S représentant la relation successeur S, elle est définie comme la relation $\{(0,1);(1,2),...,(n-2,n-1)\}$ où les sommets de G sont $\{0,1,...,n-1\}$. On définit, à partir de S, deux formules à une variables libres : $\zeta(x) = \forall y \neg S(y,x)$ qui vérifie si x est le sommet 0 et $\eta(x) = \forall y \neg S(x,y)$ qui vérifie si x est le sommet $n-1$.

Comme les variables représentent les sommets, elles sont représentés par des nombres de 0 à $n-1$, donc tout k-uplet de variables peut être vu comme un nombre entre 0 et $n^k - 1$ par l'écriture en base n : $(x_1,...,x_k)$ représente donc $\sum_{i=0}^{k-1} x_{i+1} \times n^i$. On écrit $\bar{x}$ pour le k-uplet $(x_1,...,x_k)$. On définit, à partir de S, la relation $S_k(\bar{x},\bar{y})$ qui est vérifié si $\bar{y}$ est le successeur de $\bar{x}$: S_k est la relation successeur de $\{0,...,n^k-1\}$. On définit en fait S_j par récurrence sur j. S_0 est la relation S définit ci-dessus. Si l'on connaît S_{j-1}, on définit S_j par

$$(S(x_j,y_j) \wedge (x_1 = y_1) \wedge ... \wedge (x_{j-1} = y_{j-1})) \vee (\eta(x_j) \wedge \zeta(y_j) \wedge S_{j-1}(x_1,...,x_{j-1},y_1,...,y_{j-1}))$$

On sait maintenant "compter" jusqu'à $n^k - 1$, on peut donc décrire T la table des configurations de la machine M pour une entrée de longueur n. On définit donc, pour

chaque caractère a apparaissant dans la table des configurations, un nouveau symbole de relation T_a d'arité $2k$. $T_a(\bar{x}, \bar{y})$ est la relation qui est vrai si le (i,j)ème caractère de la table des configurations est a, où $\bar{x}$ représente i et $\bar{y}$ représente j. Si l'on appelle m le nombre de symbole pouvant apparaître dans la table des configurations (la taille de l'alphabet de bande fois le nombre d'états), on crée donc m nouveaux symboles de relations $T_{a_1}, ..., T_{a_m}$. Enfin, on définit les deux derniers symboles de relations : $C_0(\bar{x})$ et $C_1(\bar{x})$. Ce sont des relations k-aire qui exprime le fait que le choix non déterministe 0 ou 1 a été fait à l'étape i où $\bar{x}$ encode i.

La formule ϕ cherchée est donc finalement $\exists S \exists T_{a_1} ... \exists T_{a_m} \exists C_0 \exists C_1 \psi$. Il ne reste maintenant plus qu'à décrire la formule ψ. ψ doit exprimer les faits suivants :

- (a) S est la relation successeur
- (b) La première ligne et la dernière colonne de T sont correctes
- (c) Les lignes suivantes de T sont bien en accord avec les transition de M
- (d) Un seul choix non déterministe est fait à chaque étape
- (e) La machine M termine et accepte

Comment peut-on exprimer ces 5 faits en logique du première ordre ?

- (a) S peut s'exprimer par la formule suivante : $\exists v_1(\forall v_2(\neg(v_2 = v_1) \Leftrightarrow (\exists v_3 S(v_2, v_3))))\wedge$ $\exists v_4(\forall v_2(\neg(v_2 = v_4) \Leftrightarrow (\exists v_3 S(v_3, v_2))))$. La formule ne fait qu'exprimer les propriétés de la relation successeur : tout le monde est successeur de quelqu'un sauf un élément, et que tout le monde admet un successeur sauf un élément.
- (b) Il faut se rappeler de la convention sur l'entrée de la machine de Turing. Si $\bar{x}$ encode 0 alors on distingue deux cas : soit les $k-2$ dernières composantes de $\bar{y}$ sont toutes nulles (ce qu'on le peut savoir grâce à la relation ζ) et dans ce cas on a $G(x_1, x_2) \Rightarrow T_1(\bar{x}, \bar{y}) \wedge \neg G(x_1, x_2) \Rightarrow T_0(\bar{x}, \bar{y})$ où $\bar{x}$ est le k-uplet $(x_1, ..., x_k)$, soit les $k-2$ dernières composantes de $\bar{y}$ ne sont pas toutes nulles et dans ce cas on a $T_\sharp(\bar{x}, \bar{y})$.
- (c) Remarquons tout d'abord que l'état de la table des configurations au temps $i > 0$ ne dépend que de la table des configurations au temps $i-1$ et plus précisément $T(i,j)$ ne dépend que de $T(i-1, j-1)$,$T(i-1, j)$ et $T(i-1, j+1)$. En effet, soit aucun de 3 éléments au dessus dans la table des configurations est de la forme a_q où a est une lettre de l'alphabet de bande et q un état et donc la bande ne peut changer entre le temps $i-1$ et i, soit un des 3 est de cette forme et alors le contenu peut peut-être changer. Une transition peut donc être vu comme un quintuplet $(\alpha, \beta, \gamma, c, \delta)$ où $\alpha = T(i-1, j-1)$, $\beta = T(i-1, j)$, $\gamma = T(i-1, j+1)$, c représente le choix non déterministe fait au temps $i-1$ et $\delta = T(i,j)$. Pour chacun des quintuplets que l'on peut construire à partir de la machine M on ajoute la formule suivante :

  $$(S_k(\bar{x'}, \bar{x}) \wedge S_k(\bar{y'}, \bar{y}) \wedge S_k(\bar{y}, \bar{y''}) \wedge T_\alpha(\bar{x'}, \bar{y'}) \wedge T_\beta(\bar{x'}, \bar{y}) \wedge T_\gamma(\bar{x'}, \bar{y''}) \wedge C_c(\bar{x'})) \Rightarrow T_\delta(\bar{x}, \bar{y})$$

- (d) A chaque étape on ne fait qu'un choix déterministe, ceci s'exprime facilement par $(C_O(\bar{x}) \vee C_1(\bar{x})) \wedge (\neg C_O(\bar{x}) \vee \neg C_1(\bar{x}))$
- (e) La machine M accepte seulement si l'état sur la dernière ligne est q_+. Si $\bar{x}$ encode $n^k - 1$, il faut voir s'il existe $\bar{y}$ tel que $T_{a_{q_+}}(\bar{x}, \bar{y})$ soit vrai pour au moins un a. On ajoute donc l'expression $\exists \bar{y} T_{a_1 q_+}(\bar{x}, \bar{y}) \vee ... \vee T_{a_p q_+}(\bar{x}, \bar{y})$ où p est le nombre de lettres de l'alphabet de bande.

On fait la conjonction des expressions définies en (a),(b),(c),(d) et (e) et on les quantifie par $5k$ quantificateurs universels pour $\bar{x}$, $\bar{x'}$, $\bar{y}$, $\bar{y'}$ et $\bar{y''}$.

Il est maintenant facile de vérifier qu'un graphe G est un modèle de ϕ si et seulement si G est accepté par la machine de Turing M, ie si et seulement si $G \in \mathcal{G}$. □

Conclusion

Le problème qui consiste à déterminer s'il existe un chemin entre deux sommets d'un graphe n'est pas exprimable en logique du première ordre (remarque 1.9) et est dans **P** , cela montre bien qu'on ne peut pas avoir la réciproque du théorème 2.3 qui relierait **P** et la logique du première ordre. Mais il existe d'autre type de logique du second ordre (citons par exemple la logique de Horn) et d'autres résultats liant un type de logique et des classes de complexités ou de langages.

Références

[1] C. Papadimitriou. *Computational complexity.* Addison-Wesley, 1995.