Inclusion et Équivalence de Requêtes Conjonctives

Stage de M2 réalisé sous la direction de Luc Segoufin

Benoît GROZ

Introduction

L'essor de l'informatique depuis les années 1950 a entraîné la nécessité d'organiser et de gérer de grandes quantités d'informations. Ceci a donné naissance aux bases de données, une *base de données* étant un ensemble organisé et intégré de données. Le système de gestion de base de données est le logiciel chargé d'organiser et de manipuler ces données. Il doit assurer un accès optimal aux données, donc le moins coûteux en temps de calcul et en espace mémoire. La recherche d'une information dans la base de données se fait en interrogeant la base au moyen de *requêtes*. Il est donc crucial que cette extraction de données se fasse le plus efficacement possible. On cherche ainsi à formuler la requête aussi intelligemment que possible : c'est l'optimisation de requête. Le travail dont ce rapport rend compte se place dans ce cadre de l'optimisation des requêtes. Les années 1980 ont vu la naissance du modèle relationnel pour les bases de données, proposé par le chercheur d'IBM. E.F.Codd, et qui représente intuitivement les données sous forme de tableaux. Ce modèle a donné lieu à une norme, qui définit un langage de description des tableaux, mais aussi un langage de requêtes (SQL).

La majorité des requêtes utilisées en pratique sont des *requêtes conjonctives*, c'est-à-dire une conjonction de prédicats positifs, auxquels nous pouvons ajouter selon le modèle étudié des prédicats d'inégalités, les variables étant quantifiées existentiellement. Ainsi, si on dispose d'une base de données contenant des prédicats *Père(x,y)* pour représenter que x est père de y, et qu'on veut connaître les paires *(g, f)* où g est le grand-père de f, on utilisera une requête de la forme : $Q(g, f) : -\exists z \cdot Pere(g, z) \wedge Pere(z, f)$. Il apparaît que les conjonctions sont particulièrement coûteuses, puisqu'elles correspondent aux jointures dans le modèle relationnel. Pour optimiser une requête conjonctive, on cherche donc à obtenir une requête conjonctive équivalente en minimisant son nombre de conjonctions[CM93]. Ceci impose de pouvoir tester l'équivalence de requêtes conjonctives, c'est-à-dire de savoir quand le résultat rendu par deux requêtes est le même. Une approche courante est de considérer un problème plus général : l'inclusion de requêtes conjonctives.

Ce travail a déjà été mené dans le cadre d'une sémantique dite *ensembliste*. La sémantique ensembliste considère le résultat d'une évaluation comme un ensemble de n-uplets, et élimine donc les doublets dans les calculs intermédiaires. Mais cette sémantique ne permet pas de représenter des requêtes usuelles utilisant des fonctions d'aggrégation comme $COUNT$ pour compter le nombre de fois qu'un élément apparaît dans le résultat d'une requête, ou $AVERAGE$ pour faire des moyennes. Ceci justifie l'étude d'une sémantique prenant en compte ces aspects : la sémantique multiensembliste. La sémantique multiensembliste considère le résultat d'une requête conjonctive comme un multiensemble, ainsi on garde les copies d'un même n-uplet. Dans le cadre de ce stage, j'ai donc étudié l'inclusion et l'équivalence de requêtes conjonctives en sémantique multiensembliste.

Les problèmes d'inclusion et d'équivalence sont de nature très différente en sémantique ensembliste et multiensembliste. Ainsi, l'inclusion de requêtes conjonctives avec inégalités est Π_2^p-complète en sémantique ensembliste [vdM97] alors qu'elle est indécidable en sémantique multiensembliste [JKV06]. De même pour les unions de requêtes conjonctives [SY80] [IR95]. L'inclusion comme l'équivalence de requêtes conjonctives sans inégalités se ramène au test d'homorphisme de graphes en sémantique ensembliste, alors que l'équivalence en sémantique multiensembliste se ramène à l'isomorphisme de graphes [CV93]. L'équivalence de requêtes conjonctives avec inégalités est, elle, dans $PSPACE$ [NSS98], par contre sa complexité précise reste inconnue. Il y a peu de résultats sur l'inclusion de requêtes en sémantique multiensembliste.

Au cours de ce stage, nous avons proposé un algorithme pour tester plus efficacement l'équivalence de requêtes conjonctives. Notre travail montre que lorsqu'il y a peu ou beaucoup d'inégalités, le problème de l'équivalence de requêtes se ramène à l'isomorphisme de graphe. Ainsi, seule la situation intermédiaire pose encore problème. Par ailleurs nous avons aussi étudié le problème d'inclusion de requêtes conjonctives sans inégalités, et nous apportons des éléments de réponse dans le cas de requêtes acycliques.

1 Les Requêtes Conjonctives

1.1 Définitions

Définition. *Un domaine est un ensemble fini de constantes. Une* relation *R d'arité k sur le domaine $\mathcal{C} = \{c_1, ...c_n\}$ est un (multi)ensemble de k-uples. Un* schéma *de base de données est un ensemble de symboles de relations $R_1, R_2, ..R_m$ auxquels on attribue une arité $k_1, k_2, ...k_m$ fixée.*

Une instance de base de données*, ou plus simplement une* base de données *sur le schéma $\mathcal{S}$ est la donnée d'un domaine fini $\mathcal{C}$ et, pour tout symbole de relation R de $\mathcal{S}$ d'arité k, d'une relation R sur $\mathcal{C}$ d'arité k.*

Notations. *D une base de données. Pour toute relation R dans D, et tout k-uple $(a_1, ...a_k)$, on notera $|R(a_1, ...a_k)|_D$ le nombre de faits $(a_1, \ldots, a_k)$ dans la relation R, dans la base de données D. Il prend donc les valeurs 0 ou 1 dans le cadre de ce rapport.*

Par ailleurs, on notera souvent $\vec{x}$ un vecteur (ou un tuple) de variables $(x_1, .., x_n)$. Et pour toute fonction f, la notation $f(\vec{x})$ représente le tuple $(f(x_1), ..., f(x_n))$. Enfin, on omettra de mentionner le domaine et le schéma chaque fois qu'ils ne jouent aucun rôle ou sont évidents.

Une requête est une fonction qui prend en entrée une base de données (c-à-d. un ensemble de relations) et retourne une relation :

Définition. *Une* Requête conjonctive *(CQ) est une règle de la forme :*

$$Q(x_1, \ldots x_k) : -T_1(\vec{z_1}), \ldots, T_n(\vec{z_n})$$

où les variables des tuples $\vec{z_i}$ peuvent être de deux types différents :

- *soit elles sont un des x_i, auquel cas on les dira* libres
- *soit elles ne sont pas dans $\vec{x}$ (on écrira ces variables y_i), auquel cas on les dira* liées*, ce qui signifiera qu'elles sont implicitement quantifiées existentiellement.*

Les faits $T_i(\vec{z_i})$ constituent le *corps* de la requête.

Soit $Q(\vec{x}) : -T_1(\vec{z_1}), \ldots, T_n(\vec{z_n})$. Une *valuation* est une application τ des variables de Q dans le domaine de D.

Définition. *Une valuation τ sera appelée* homomorphisme de Q vers D *si pour tout $T_i(\vec{z_i})$ dans le corps de Q, le fait $T_i(\tau(\vec{z_i}))$ est présent dans D. On notera ainsi $\tau(T_i(\vec{z_i}))$ le fait $T_i(\tau(\vec{z_i}))$.*

Exemple 1. *Considérons la base de données cinématographique suivante :*

La base de données D	
Film(F.Capra, La vie est belle)	Acteur(J.Stewart, La vie est belle)
Film(F.Capra, Mr Smith au Sénat)	Acteur(J.Stewart, Mr Smith au Sénat)
Film(A. Hitchcock, Fenêtre sur cour)	Acteur(J. Stewart, Fenêtre sur cour)
	Acteur(G. Kelly, Fenêtre sur cour)

et la requête :

$$Q(aut0, act0) : -Film(aut0, tit0), Acteur(act0, tit0)$$

Les homomorphismes de Q dans D sont les h_i donnés par le tableau suivant :

	h_1	h_2	h_3	h_4
aut0	F.Capra	F.Capra	A. Hitchcock	A. Hitchcock
act0	J.Stewart	J.Stewart	J.Stewart	G. Kelly
tit0	La vie est belle	Mr Smith au Sénat	Fenêtre sur cour	Fenêtre sur cour

FIG. 1 – Homomorphismes de Q dans D.

On considère aussi le cas des requêtes avec inégalités. Une requête avec inégalités ($CQ^{\neq}$) est un couple composé d'une requête $CQ : Q$, et d'un ensemble d'inégalités $\mathcal{N}$ sur les variables de Q (qu'on considèrera généralement comme un ensemble de couples). Étant données une base D et une requête avec inégalités $(Q, \mathcal{N})$, on appellera *homomorphisme* de $(Q, \mathcal{N})$ dans D tout homomorphisme h de Q dans D qui respecte les inégalités de $\mathcal{N}$, c-à-d. tel que $\forall (x, x') \in \mathcal{N} \cdot h(x) \neq h(x')$.

Enfin, il faut noter que le corps d'une requête Q' est un ensemble de faits $T_i(\vec{z_i})$, donc on peut par abus le considérer comme une base de données composée de ces faits, les constantes du domaine étant les variables apparaissant dans $Q' : D = \{T_i(\vec{z_i})|i\}$. C'est ce qui explique que l'on parle d'homomorphisme d'une requête Q dans le corps d'une autre requête Q'.

1.2 Sémantiques

1.2.1 Sémantique ensembliste

Il nous reste à définir le résultat de l'évaluation d'une requête CQ sur une base de données. En sémantique ensembliste les relations sont des ensembles de faits (et les bases de données aussi), et l'évaluation d'une requête rend un ensemble de faits.

Définition. *Soit* $Q(\vec{x}) : -T_1(\vec{z_1}), \ldots, T_n(\vec{z_n})$ *une requête, et* D *une base de données.*

$$Eval_S(Q, D) = \{(a_1, \ldots, a_k) | \exists \tau \text{ homomorphisme de } Q \text{ vers } D \text{ tel que } \tau(\vec{x}) = (a_1, \ldots, a_k)\}$$

Exemple : Reprenons l'exemple 1. Dans cet exemple, évaluer la requête Q sur la base D rend les couples (cinéaste,acteur) qui ont collaboré sur un film :

$$Eval_S(Q, D) = \{(F.Capra, J.Stewart), (A.Hitchcock, J.Stewart), (A.Hitchcock, G.Kelly)\}$$

Exemple avec inégalités : Si on pose $Q(aut0) : -Film(aut0, tit0), Film(aut0, tit0')$ et $\mathcal{N} = \{tit0 \neq tit0'\}$, la requête $(Q, \mathcal{N})$ sélectionne les réalisateurs qui ont réalisés au moins deux films.

Définition. *Conjunctive Query Containment : Etant données deux requêtes* Q *et* Q'*, nous disons que* Q *est contenue dans* Q' *en sémantique ensembliste (*$Q \leq_S Q'$*) ssi*

$$\forall D \cdot Eval_S(Q, D) \subseteq Eval_S(Q', D)$$

i.e. si $\forall D, \vec{X} \cdot \vec{X} \in Eval_S(Q, D) \implies \vec{X} \in Eval_S(Q', D)$

et la définition pour Conjunctive query equivalence en découle :

Définition. *Conjunctive Query Equivalence : Deux requêtes* Q *et* Q' *sont* équivalentes *en sémantique ensembliste(*$Q \equiv_S Q'$*) ssi*

$$Q \leq_S Q' \text{ et } Q' \leq_S Q$$

i.e. si $\forall D, \vec{X} \cdot \vec{X} \in Eval_S(Q, D) \Leftrightarrow \vec{X} \in Eval_S(Q', D)$

Ceci définit la sémantique ensembliste. Cependant la sémantique ensembliste ne permet pas des opérations de comptage (on ne peut pas compter exactement combien de films un réalisateur a tourné), alors que les requêtes utilisées dans la réalité utilisent ce genre d'opérations. Ceci a amené les informaticiens à étudier une sémantique basée sur des multiensembles plutôt que des ensembles([CV93], [IR95]).

1.2.2 Sémantique multiensembliste

Pour la sémantique multiensembliste on définit normalement une base de données comme un multiensemble de faits, et évaluer une requête retourne un multiensemble de faits.

Définition.

$$Eval_B(Q, D)(a_1, \ldots, a_k) = \sum_{\tau \in \mathcal{A}} \prod_{i=1}^{n} |T_i(\tau(\vec{z_i}))|_D$$

où

- $\mathcal{A}$ *est l'ensemble des homomorphismes de* Q *vers* D *qui envoient* x_j *sur* a_j *pour tout* $j \in [1..k]$.
- $T_1(\vec{z_1}), ...T_n(\vec{z_n})$ *le corps de* Q.
- *pour tout* x, $|T_i(x)|_D$ *vaut* 1 *si* $T_i(x)$ *est un fait de* D, 0 *sinon.*

$Eval_B(Q,D)(a_1,\ldots,a_k)$ compte donc simplement le nombre d'homomorphismes dans $\mathcal{A}$. Exemple : Dans l'exemple 1 on peut être intéressé par compter le nombre de films sur lesquels tel acteur et tel réalisateur ont collaboré. C'est ce que donne le multiensemble suivant (où la multiplicité d'un tuple est indiquée entre crochets) :

$$Eval_B(Q,D) = \{(F.Capra, J.Stewart)[2], (A.Hitchcock, J.Stewart)[1], \\ (A.Hitchcock, G.Kelly)[1]\}$$

Nous pouvons maintenant définir l'inclusion et l'équivalence de façon naturelle :

Définition. *Conjunctive Query Containment ($CQCon_B$) : Etant données deux requêtes Q et Q', Q est contenue dans Q' en sémantique multiensemble ($Q \leq_B Q'$) ssi*

$$\forall D, (a_1,\ldots,a_n) \cdot Eval_B(Q,D)(a_1,\ldots,a_n) \leq Eval_B(Q',D)(a_1,\ldots,a_n)$$

et la définition pour Conjunctive query equivalence est donc :

Définition. *Conjunctive Query Equivalence ($CQEq_B$) : Deux requêtes Q et Q' sont* équivalentes *en sémantique multiensemble ($Q \equiv_B Q'$) ssi*

$$Q \leq_B Q' \text{ et } Q' \leq_B Q$$

i.e. si $\forall D, (a_1,\ldots,a_n) \cdot Eval_B(Q,D)(a_1,\ldots,a_n) = Eval_B(Q',D)(a_1,\ldots,a_n)$

Exemple 2. *Prenons un exemple donné par* [CV93]. *Considérons les requêtes :*
$Q_1(x,y) :- R(x), T(y), S(z,x), S(t,y)$ *et*
$Q_2(x,y) :- R(x), T(y), S(z,t), S(w,t)$.
On peut vérifier que $Q_1 \leq_B Q_2$. [1]

Nous allons maintenant traduire le problème d'inclusion de requêtes en termes d'homomorphismes de graphes. Pour cela, nous introduisons un nouveau formalisme que nous serons amenés à utiliser par la suite.

1.3 Des requêtes conjonctives aux homomorphismes de graphes

Un schéma de graphe $\mathcal{S}$ est la donnée d'un ensemble de symboles d'hyperarêtes $\{E_1, ..., E_m\}$ avec une arité fixée : $k_1, ...k_m$.

Définition. *Un* hypergraphe *sur le schéma $\mathcal{S} = \{E_1, ...E_m\}$ est la donnée*
- *d'un ensemble V_G de sommets*
- *d'un ensemble fini $\{E_1(G), E_2(G), ...E_m(G)\}$ où chaque $E_i(G)$ est un ensemble d'hyperarête d'arité k_i. Ces hyperarêtes (des k_i-uplets) généralisent les arêtes que l'on représenterait comme des couples.*

Par abus on parlera parfois de graphes au lieu d'hypergraphes.

On définit alors de façon naturelle les homomorphismes de graphes :

Définition. *Soient G et H deux hypergraphes sur le schéma $\mathcal{S} = E_1, \ldots, E_m$. Une fonction $\phi : V_G \mapsto V_H$ est un* homomorphisme *si pour tout $i \in [1..m]$, pour tout $(a_1, a_2, ...a_{k_i}) \in E_i(G)$, on a $(\phi(a_1), \phi(a_2), ...\phi(a_{k_i})) \in E_i(H)$.*

[1] cf preuve en annexe

Un homomorphisme ϕ est dit *injectif* si pour tous $v, v' \in V_G$, $\phi(v) \neq \phi(v')$. Un homomorphisme est dit *surjectif* si pour tout $i \in [1..m]$, pour tout $(b_1, b_2, ...b_{k_i}) \in E_i(H)$, il existe $(a_1, a_2, ...a_{k_i}) \in E_i(G)$ tel que $(\phi(a_1), \phi(a_2), ...\phi(a_{k_i})) = (b_1, b_2, ...b_{k_i})$. Un homomorphisme à la fois surjectif et injectif est bien sûr appelé *isomorphisme*. Enfin, un *automorphisme* est un isomorphisme d'un hypergraphe dans lui-même.

Pour finir, nous pouvons étendre la traduction au cas où on autorise des inégalités dans les requêtes : soit G un graphe et $\mathcal{N}$ un ensemble de contraintes d'inégalités sur G (un ensemble de couples de sommets de G, qu'on peut représenter par un graphe non orienté sur V_G). Soit X un graphe. Un homomorphisme ϕ de G vers X *respecte les inégalités de* $\mathcal{N}$ si pour tous $v, v' \in V_G$, $\{v, v'\} \in \mathcal{N} \implies \phi(v) \neq \phi(v')$.

Notations. *On notera $hom(G, X)$ le nombre d'homomorphismes de G vers X et $hom_{\mathcal{N}}(G, X)$ le nombre d'homomorphismes de G vers X qui respectent les inégalités de $\mathcal{N}$. De même on notera $surj(G, X)$ le nombre de surjections de G vers X...*

Notations. *Etant donnés deux graphes G et H avec des ensembles de sommets disjoints, on notera $G \bigoplus H$ l'union disjointe des deux graphes, définie de façon évidente (en renommant les sommets de H si besoin).*
On utilise la notation $G \simeq H$ lorsque deux graphes G et H sont isomorphes. Enfin on utilise le symbole $\circ$ pour représenter la composition d'homomorphismes : $(f \circ g)(x) = f(g(x))$.

Définition. *Etant donnée une requête conjonctive sur le schéma $\{E_1, \dots E_m\}$*

$$Q(\vec{x}) : -T_1(\vec{z}_1), T_2(\vec{z}_2) \dots T_n(\vec{z}_n)$$

nous construisons un hypergraphe $G[Q]$ sur le schéma $E_0, E_1, \dots, E_m$ avec pour tout $i \geq 1$ E_i d'arité k_i dans Q, et E_0 d'arité la dimension de $\vec{x}$, de la façon suivante :
- *les sommets de $G[Q]$ sont les variables qui apparaissent dans la requête*
- *$(u_1, ...u_{k_i}) \in E_i$ dans $G \Leftrightarrow E_i(u_1, ..., u_{k_i})$ apparaît dans le corps de Q*
- *$E_0 = \{\vec{x}\}$.*

Le graphe ainsi construit sera appelé graphe de la requête Q.

De même, nous pouvons construire le graphe G d'une base de données D comme suit :
- les sommets sont les variables qui apparaissent dans la requête
- pour tout symbole E_i d'arité k_i, on ajoute un symbole d'hyperarête E_i et $(c_1, ...c_{k_i}) \in E_i$ dans $G \Leftrightarrow E_i(c_1, ..., c_{k_i})$ est présent dans D.

Propriétés 1. *Etant données deux requêtes conjonctives avec inégalités $(Q_1, \mathcal{N}_1)$ et $(Q_2, \mathcal{N}_2)$ sur un schéma $\mathcal{S}$, notons $\mathcal{S}_G$ le schéma de $G[Q_1]$ ou $G[Q_2]$.*
$(Q_1, \mathcal{N}_1) \leq_B (Q_2, \mathcal{N}_2)$ ssi pour tout hypergraphe X sur le schéma $\mathcal{S}_G$,
$hom_{\mathcal{N}_1}(G[Q_1], X) \leq hom_{\mathcal{N}_2}(G[Q_2], X)$

Corollaire. *Etant données deux requêtes conjonctives Q_1 et Q_2,*

$$Q_1 \equiv_B Q_2 \Leftrightarrow \forall X \cdot hom_{\mathcal{N}_1}(G[Q_1], X) = hom_{\mathcal{N}_2}(G[Q_2], X)$$

2 Les résultats en sémantique ensembliste

Soient Q, Q' deux requêtes CQ. Il est facile de montrer que $Q \leq_S Q'$ ssi il existe un homomorphisme du corps de Q' dans celui de Q([CM93]). Ceci implique que en sémantique ensembliste, *Conjunctive Query Containment* est $NP-complet$. Dès que l'on rajoute des inégalités dans les requêtes, par contre, ce problème devient Π_2^p-complet[vdM97](De même lorsque l'on considère l'inclusion d'unions de requêtes conjonctives le problème est aussi Π_2^p-complet[CM93]). L'équivalence de requêtes CQ est $NP-$complète.

3 Conjunctive Query Containment en sémantique multiensembliste

En sémantique multiensembliste, les choses se compliquent. Pour des requêtes avec inégalités, Kolaitis, Vee et Majumdar[JKV06] ont montré que le problème devenait indécidable par réduction du 10^e problème de Hilbert. De même, Ioannidis et Ramakrishnan[IR95] ont montré que le problème de l'inclusion d'unions de requêtes conjonctives était indécidable[2]. On n'a pas de résultats sur la complexité de *Conjunctive Query Containment*. On sait ce problème $NP-$dur, car on y réduit facilement le problème 3-coloriabilité. Mais on ne sait même pas si il est décidable. Par contre, on dispose d'une série de conditions suffisantes ou nécessaires [CV93][IR95].

Soient Q et Q' deux requêtes conjonctives sur un schéma $\mathcal{S}$. Avec notre définition pour la sémantique multiensembliste, on peut supposer sans perte de généralité que le corps des requêtes Q et Q' ne contient pas deux fois le même prédicat répété avec les mêmes variables en même position $T_1(z_1, \ldots z_n), T_1(z_1, \ldots z_n)$ (si c'était le cas on pourrait éliminer les doublets dans le corps de la requête sans modifier le résultat de l'évaluation ...).

Propriétés 2. *Avec les conventions ci-dessus,*

- *Si $Q \leq_B Q'$ alors pour tout symbole T_i dans $\mathcal{S}$, le nombre d'occurrences de T_i dans le corps de Q' est au moins égal au nombre d'occurrences de T_i dans le corps de Q.*
- *Si $Q \leq_B Q'$ alors pour tout fait $T_i(\vec{z_i})$ dans le corps de Q, il existe un homomorphisme τ de Q' dans le corps de Q tel que $T_i(\vec{z_i})$ est l'image par τ d'un fait de Q'.*
- *S'il existe un homomorphisme de Q' dans le corps de Q surjectif alors $Q \leq_B Q'$.*
- *Si tout symbole T_i de $\mathcal{S}$ apparaît au plus une fois dans Q, alors $Q \leq_B Q'$ si et seulement si il existe un homomorphisme surjectif de Q' dans le corps de Q.*

Comme le fait remarquer [CV93], ces propriétés font apparaître une différence de fond entre sémantique ensembliste et multiensembliste. En effet, pour que $Q \leq_S Q'$, il faut que le corps de Q soit plus contraignant que celui de Q', donc moins il y a de faits dans Q' plus il est facile d'avoir $Q \leq_S Q'$. Alors qu'en bag-sémantique, peu de conjonctions implique peu de combinaisons possibles, donc une faible multiplicité, ce qui rend plus dur d'obtenir $Q \leq_B Q'$.

Une série de papiers récents ([BH97] et des développements) soutenait que pour tester si $Q \leq_B Q'$, il suffisait de considérer un nombre fini de familles de bases de données (grosso modo : les sous parties du corps de Q). Mais Vee a montré que cette affirmation était erronnée[Vee06] (l'erreur de Brisaboa vient de ce qu'il faut considérer des unions non disjointes de "copies" isomorphes de sous parties du corps de Q. Il n'est plus aussi facile de représenter finiment toutes les possibilités). On n'a donc pas de caractérisation intéressante de l'inclusion de requêtes en

[2] cf Annexe

sémantique multiensembliste (en particulier on ne sait pas borner la taille des bases de données à considérer comme dans le théorème 4 pour l'équivalence).

Ceci nous a amené à considérer une classe restreinte de requêtes, dans l'espoir que $CQCon_B$ soit plus facile à caractériser sur cette classe. Ainsi, nous nous sommes restreints aux requêtes dont le graphe associé est un arbre. Nous montrons que même sur cette classe très restreinte $CQCon_B$ n'est pas un problème trivial.

Dans la suite de cette section, lorsque T et T' sont deux arbres nous noterons par abus de notation $T \leq_B T'$ pour signifier que pour tout graphe X, $hom(T, X) \leq hom(T', X)$. De façon naturelle, T et T' seront dits incomparables si l'on n'a ni $T \leq_B T'$ ni $T' \leq_B T$.

Problème : Si T, T' arbre orientés, quand a-t-on $T \leq_B T'$?

3.1 Généralités, propriétés élémentaires sur les arbres

Définition. *Nous appelons* arbres *des graphes orientés acycliques. L'orientation choisie ici est que les arêtes vont de la racine vers les feuilles.*

Définition. *La profondeur de la racine est* 1*, et la profondeur d'un noeud est définie récursivement comme la profondeur de son père plus un.*
La hauteur $h(T)$ *d'un arbre T est la profondeur de sa feuille la plus éloignée de la racine.*

On appelle enfin "fil" de longueur k l'arbre de hauteur k dont tous les noeuds sont d'arité 1 (sauf la feuille).

Commençons par donner quelques propriétés simples que doivent vérifier deux arbres pour qu'il y ait inclusion. Une première propriété traduit le fait que si un arbre T est en un certain sens "inclus" dans un autre arbre T', alors $T \leq_B T'$.

Propriétés 3. *Soit T un arbre. T' obtenu à partir de T en ajoutant à un sommet quelconque v dans T un sous arbre dont la longueur ne dépasse pas celle de la plus longue branche issue de ce noeud. Alors $T \leq_B T'$*

Démonstration. En effet pour tout graphe X, on peut associer de façon injective un morphisme de T' dans X à tout morphisme de T dans X. Il suffit de choisir une branche de longueur maximale l, puis, pour tout noeud à profondeur $j \leq l$ dans T' on envoie ce noeud sur l'image du noeud à profondeur j dans la branche la plus longue. □

Il est bon de noter que par transitivité de $\leq_B$, on peut appliquer plusieurs "ajouts", ce qui rend cette propriété très pratique pour traiter les cas triviaux. Elle ne donne cependant pas une condition nécessaire comme nous le verrons en étudiant le cas des arbres de profondeur 2. Une propriété remarquable de $CQCon_B$ sur les arbres est que deux arbres de hauteur différente sont incomparables.

Propriétés 4. $T \leq_B T' \implies T$ *et* T' *sont de même hauteur.*

Démonstration. Soient T et T' deux arbres tels que $T \leq_B T'$.
Premièrement on montre que $h(T') \leq h(T)$.

En effet, si $h(T') > h(T)$, on considère X le fil de longueur $h(T)$: $hom(T, X) = 1 > hom(T', X) = 0$, ce qui contredit nos hypothèses.

De manière moins évidente, on montre aussi que $h(T) \leq h(T')$: En effet, supposons $h(T) > h(T')$. Posons $l = h(T)$, $l' = h(T')$ et n l'arité maximale d'un noeud de T'. Prenons

pour X la famille des graphes $X_{k,m}$ représentée dans la figure suivante : tous les noeuds sont d'arité entrante m.

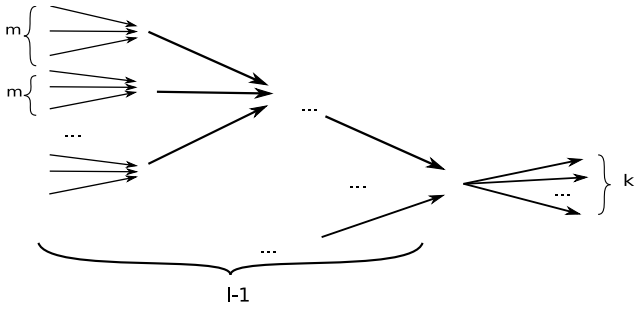

FIG. 2 – arbre $X_{k,m}$

Alors, par la propriété 3, le fil T_1 de longueur l vérifie $hom(T_1, X_{k,m}) \leq hom(T, X_{k,m})$ pour tous k, m. De même, soit T_1' l'arbre complet de profondeur l' dont tous les noeuds sont d'arité n. On a $hom(T', X_{k,m}) \leq hom(T_1', X_{k,m})$ toujours par la propriété 3

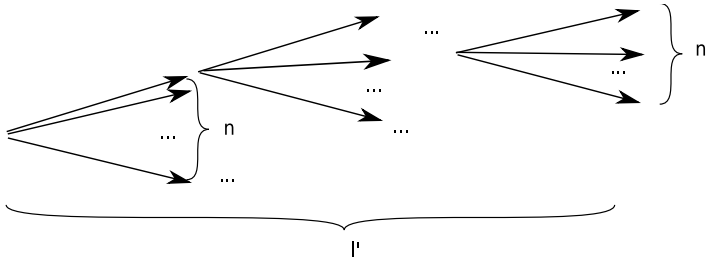

FIG. 3 – arbre T_1'

Or $hom(T_1, X_{k,m}) = m^{l-1} * k$,
et $hom(T_1', X_{k,m}) = m^{l'-1} * k^n + m^{l'} * \frac{m^{l-l'}-1}{m-1} = m^{l'-1}k^n + \frac{m^l - m^{l'}}{m-1}$. Comme pour tous l, n et tout $l' < l$ il existe k, m tel que $m^{l-1} * k > m^{l'-1}k^n + \frac{m^l - m^{l'}}{m-1}$ (prendre par exemple, $k = 3$ et $m > k^n$), on en déduit qu'il existe un graphe $X_{k,m}$ vérifiant :
$hom(T', X_{k,m}) \leq hom(T_1', X_{k,m}) < hom(T_1, X_{k,m}) \leq hom(T, X_{k,m})$. Ceci contredit l'hypothèse $T \leq_B T'$. Ainsi, la hauteur de l'arbre T ne peut dépasser celle de T'. Ce qui conclut la preuve. □

Après avoir donné une condition nécessaire sur la hauteur des arbres, donc une condition "verticale", nous pouvons donner une condition plus "horizontale" pour l'inclusion d'arbres.

Propriétés 5. *Soient T, T' deux arbres, notons N_k le nombre de branches de hauteur exactement k dans T. et N_k' le nombre de branches de hauteur exactement k dans T'. Alors $T \leq T' \implies N_k \leq N_k'$ pour tout k.*

Démonstration. On considère la famille de graphe X_k représentée sur la figure suivante.

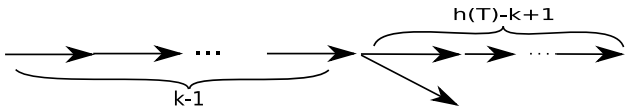

FIG. 4 – l'arbre X_k

$hom(T, X_k) = 2^{N_k}$. Ce qui donne directement le résultat. □

3.2 Vers la décidabilité ?

Pour tester l'inclusion de deux requêtes Q_1 et Q_2, on doit comparer l'évaluation de ces requêtes sur toute base de données D, c'est-à-dire lorsque l'on modélise le problème avec des graphes que l'on doit comparer le nombre d'homomorphismes des graphes $G[Q_i]$ dans X pour tout graphe X. Un des résultats très satisfaisant de cette section est que lorsque les requêtes sont des arbres T et T', on peut restreindre les graphes X aux arbres, et même, puisque $T \leq_B T' \implies h(T) = h(T')$, aux arbres de hauteur $h(T)$.

Notations. *Soit X un graphe et v un sommet de X. Pour tout graphe T et tout sommet r dans T , on notera $\overset{hom(T,X)}{\underset{r\mapsto v}{}}$ le nombre d'homomorphismes de T dans X qui envoient le sommet r sur v.*

Théorème 1. *Soit k un entier. Pour tout graphe X_0 et tout sommet v de X_0, soit il existe un arbre X' de hauteur k tel que pour tout arbre T de hauteur k et de racine r, $\overset{hom(T,X_0)}{\underset{r\mapsto v}{}} = hom(T, X')$ soit pour tout arbre T de hauteur k et de racine r, $\overset{hom(T,X_0)}{\underset{r\mapsto v}{}} = 0$.*

Démonstration. On prend pour X' le dépliage de X_0, que l'on coupe à profondeur k. Si ce dépliage est de profondeur $h < k$ aucun arbre de profondeur k ne peut s'y plonger. Sinon il est clair qu'il y a correspondance entre les homomorphismes de T dans X_0 qui envoient r sur v et les homomorphismes de T dans X' (qui envoient tous forcément v sur la racine de X'). □

Propriétés 6. *Soient T et T' deux arbres de hauteur k, $T \leq_B T'$ ssi pour tout* arbre X *de hauteur k, $hom(T, X) \leq hom(T', X)$.*

Démonstration. $\Rightarrow$: Si $T \leq_B T'$ alors il est évident que pour tout *arbre* X de hauteur k, $hom(T, X) \leq hom(T', X)$.

$\Leftarrow$: Soient T et T' des arbres de racine r et r'. Si pour tout *arbre* X de hauteur k, $hom(T, X) \leq hom(T', X)$, alors considérons un graphe quelconque X_0. Par le théorème 1, pour tout sommet v de X_0, il existe un arbre X de hauteur k tel que $\overset{hom(T,X_0)}{\underset{r\mapsto v}{}} = hom(T, X)$ et $\overset{hom(T',X_0)}{\underset{r'\mapsto v}{}} = hom(T', X)$. Par hypothèse, ceci implique que pour tout sommet v de X_0, $\overset{hom(T,X_0)}{\underset{r\mapsto v}{}} = hom(T, X) \leq hom(T', X) = \overset{hom(T',X_0)}{\underset{r'\mapsto v}{}}$. Par suite, $hom(T, X_0) \leq hom(T', X_0)$ □

Propriétés 7. *On peut maintenant raffiner la propriété 5 : Soit G un arbre de hauteur k. Pour tout $i \leq k$, notons G_i l'arbre G coupé à profondeur i, c'est-à-dire dont on enlève tous les noeuds à profondeur supérieure à i. Soient maintenant T et T' deux arbres de hauteur k. Si $T \leq_B T'$, alors pour tout $i \leq k$, $T_i \leq_B T'_i$.*

Démonstration. Soient T et T' deux arbres de hauteur k tels que $T \leq_B T'$. On considère pour le montrer l'ensemble $\mathcal{E}_i$ de tous les arbres X de hauteur i que l'on prolonge par des fils de longueur $k - i$.

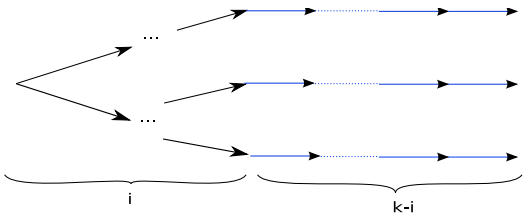

FIG. 5 – Un arbre de $\mathcal{E}_i$

Soit G un arbre de hauteur i. On le prolonge par des fils en un arbre X de hauteur k. Alors on voit facilement que $hom(T_i, G) = hom(T, X) \leq hom(T', X) = hom(T'_i, G)$. Ainsi, pour tout arbre G de hauteur i, $hom(T_i, G) \leq hom(T'_i, G)$. Par la proposition 6, ceci implique que $T_i \leq_B T'_i$. □

La propriété 6 ne donne pas a priori la décidabilité dans le cas général car l'arité des noeuds de l'arbre X n'est pas bornée. En revanche, on peut utiliser une représentation symbolique finie pour l'ensemble des arbres de profondeur k : on utilise des variables pour l'arité des noeuds comme nous allons le montrer pour les arbres de profondeur2.

3.3 Propriétés plus combinatoires

On considère ici un type d'arbre particulier : les arbres de hauteur 2.

Notations. *On va abondamment utiliser les graphes X de la forme suivante, que l'on notera $X = T_{a_1,a_2,\ldots a_m}$ pour tout $m \geq 1$:*

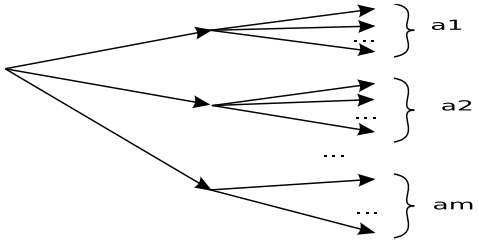

FIG. 6 – graphe X général

On supposera sauf mention explicite du contraire que les arités sont données par ordre décroissant, et que pour tout i, $a_i \geq 1$.

3.3.1 Caractérisation pour les arbres de profondeur 2 et d'arité 2 à la racine

On va d'abord s'intéresser à une classe particulière : les arbres de la forme $T_{k,l}$ (donc d'arité 2 à la racine). Soient $k, l, k', l' \geq 1$. (avec notre choix de notation, $k \geq l$ et $k' \geq l'$).
Notre but est de caractériser les cas pour lesquels $T_{k,l} \leq_B T_{k',l'}$. Par la propriété 5,
$T_{k,l} \leq_B T_{k',l'} \implies k + l \leq k' + l'$.

Étude Préliminaire On commence par étudier $hom(T_{k,l}, X)$ dans le cas où X est de la forme $T_{\alpha,\beta}$. On note $S_{k,l} = hom(T_{k,l}, T_{\alpha,\beta})$ le polynôme en α, β. On a :

$$S_{k,l} = (\alpha^k + \beta^k) \cdot (\alpha^l + \beta^l) = \alpha^{k+l} + \beta^{k+l} + \alpha^k \beta^l + \beta^k \alpha^l$$

Si $k'+l' > k+l$ alors $S_{k',l'} > S_{k,l}$. En effet, si $\alpha = \beta$ c'est évident. Sinon, supposons $\alpha > \beta$. Alors en particulier $\alpha > 2$.

$$\begin{aligned} S_{k',l'} &= \alpha \cdot \alpha^{k'+l'-1} + \beta^{k'+l'} + \alpha^{k'}\beta^{l'} + \beta^{k'}\alpha^{l'} \\ &\geq 2 \cdot \alpha^{k+l} + \beta^{k'+l'} + \alpha^{k'}\beta^{l'} + \beta^{k'}\alpha^{l'} \\ &> \alpha^{k+l} + \alpha^k\beta^l + \beta^{k+l} + \beta^k\alpha^l + 0 \\ &\quad \text{car } \alpha^{k'}\beta^{l'} = \alpha^l\beta^k\alpha^{k'-l}\beta^{l'-k} \geq \alpha^l\beta^k\beta^{k'-l}\beta^{l'-k} \geq \alpha^l\beta^k \\ &> S_{k,l}. \end{aligned}$$

Si $k'+l' = k+l$ alors $S_{k',l'} \geq S_{k,l}$ ssi $k' \geq k$. En effet, notons alors $n = k+l = k'+l'$. $S_{k,l} = \alpha^n + \beta^n + \alpha^k\beta^{n-k} + \beta^k\alpha^{n-k}$ On considère maintenant $S_{k,l}$ comme fonction de k et n, et on veut montrer que à n fixé, cette fonction croît avec k pour $n \geq k \geq n-k$.

$$\begin{aligned} \frac{\partial S_{k,l}}{\partial k} &= \ln\alpha \cdot \alpha^k\beta^{n-k} - \ln\beta \cdot \alpha^k\beta^{n-k} + \ln\beta \cdot \beta^k\alpha^{n-k} - \ln\alpha \cdot \beta^k\alpha^{n-k} \\ &= \underbrace{(\ln\alpha - \ln\beta)}_{>0} \cdot (\alpha^k\beta^{n-k} - \beta^k\alpha^{n-k}) \end{aligned}$$

Or

$$\frac{\alpha^k\beta^{n-k}}{\beta^k\alpha^{n-k}} = \left(\frac{\alpha}{\beta}\right)^{k-(n-k)}$$

Ainsi, pour $k > n-k$, $\alpha^k\beta^{n-k} > \beta^k\alpha^{n-k}$ donc $\frac{\partial S_{k,l}}{\partial k} > 0$. Par suite, $S_{k',l'} \geq S_{k,l}$ ssi $k' \geq k$.

Caractérisation de l'inclusion $T_{k,l} \leq_B T_{k',l'}$ ssi $hom(T_{k,l}, X) \leq hom(T_{k',l'}, X)$ pour tout X de la forme $T_{a_1,\ldots,a_m}$ avec $m \geq 1, a_i \geq 1$.

Soient donc $m \geq 1$ et $a_1, \ldots a_m \geq 1$ m entiers. On définit le polynôme en $a_1, \ldots a_m$: $P_{k,l} = hom(T_{k,l}, X) = (a_1^k + a_2^k + \ldots a_m^k)(a_1^l + \ldots + a_m^l)$.
Remarque : On a aussi

$$hom(T_{k,l}, X) = \sum_{1 \leq i < j \leq m} hom(T_{k,l}, T_{a_i,a_j}) - (m-2) \cdot \sum_{1 \leq i \leq m} hom(T_{k,l}, T_{a_i}) \qquad (1)$$

Vu qu'on considère des arbres d'arité deux à la racine, le plus simple est de faire une étude de cas :

Si $k+l = k'+l'$ alors supposons sans perte de généralité $k \leq k'$. En ce cas, par les résultats de la section préliminaire, $hom(T_{k,l}, T_{a_i,a_j}) \leq hom(T_{k',l'}, T_{a_i,a_j})$ pour tout i, j. De plus, pour tout i, $hom(T_{k,l}, T_{a_i}) = hom(T_{k',l'}, T_{a_i})$. D'où par l'équation 1, $P_{k,l} \leq P_{k',l'}$.

Si $k'+l' > k+l$ alors
- soit $k' \geq k$ et $l' \geq l$. Alors l'arbre $T_{k',l'}$ est l'arbre $T_{k,l}$ auquel on ajoute des branches. Clairement $P_{k,l} \leq P_{k',l'}$.
- soit $k' > k$ et $l' < l$. Alors posons $j = k+l-l'$. On a $k' > j \geq k \geq l > l'$. Ainsi $j + l' = k + l$, et $j \geq k$. Par le résultat du paragraphe précédent, on en déduit que $P_{k,l} \leq P_{j,l'}$. D'où $P_{k,l} \leq P_{j,l'} \leq P_{k',l'}$.

– soit enfin $k' < k$. En ce cas, on va montrer que $T_{k,l}$ et $T_{k',l'}$ sont incomparables. Tout d'abord, par la propriété 5, on ne peut pas avoir $T_{k,l} \geq_B T_{k',l'}$. De plus, $P_{k',l'} \leq P_{k',k'}$ et $P_{k'+1,1} \leq P_{k,l}$. Prenons $a_2 = \ldots = a_m = 1$. Alors

$$\begin{array}{lll} P_{k',k'} = & (a_1^{k'} + (m-1))^2 & = a_1^{2k'} + 2 \cdot (m-1) \cdot a_1^{k'} + (m-1)^2 \quad \text{et} \\ P_{k'+1,1} = & (a_1 + (m-1))(a_1^{k'+1} + (m-1)) & = a_1^{k'+2} + (m-1) \cdot a_1^{k'+1} + (m-1) \cdot a_1 + (m-1)^2 \end{array}$$

donc

$$P_{k',k'} - P_{k'+1,1} = -(m-1) \cdot a_1^{k'+1} + 2 \cdot (m-1) \cdot a_1^{k'} + a_1^{2k'} - a_1^{k'+2}$$

En prenant pour valeur de $(m-1)$: $a_1^{k'}$, on obtient pour $P_{k',k'} - P_{k'+1,1}$ un polynôme en a_1 de terme de plus haut degré $-a_1^{2k'+1}$. Ceci implique à son tour qu'il existe une valeur de a_1 tel que ce polynôme soit négatif.
On a ainsi prouvé que il existait des valeurs de m et de $a_1, a_2, \ldots a_m$ telles que $hom(T_{k',k'}, X) < hom(T_{k'+1,1}, X)$. D'où pour revenir à notre étude de cas $hom(T_{k',l'}, X) \leq hom(T_{k',k'}, X) < hom(T_{k'+1,1}, X) \leq hom(T_{k,l}, X)$. Ce qui conclut cette étude.

Conclusion

Théorème 2. *Soient $k, l, k', l' \geq 1$. On suppose sans perte de généralité $k \geq l$ et $k' \geq l'$. Alors*

$T_{k,l} \leq_B T_{k',l'}$	*si $k + l \leq k' + l'$ et $k \leq k'$*
$T_{k',l'} \leq_B T_{k,l}$	*si $k' + l' \leq k + l$ et $k' \leq k$*
$T_{k,l}$ *et* $T_{k',l'}$ *sont incomparables*	*si $k + l \leq k' + l'$ et $k > k'$*
	ou si $k' + l' \leq k + l$ et $k' > k$

Si on veut considérer tous les cas où l'arité de la racine est égale à deux, il faut encore traiter le cas des arbres $T_{k,l}$ où $l = 0$ (c-à-d. avec une feuille à profondeur 1). Le seul cas encore intéressant est de comparer $T_{k,0}$ avec un arbre $T_{k',l'}$ où $k' \geq l' \geq 1$.

Par la propriété 5, on ne peut jamais avoir $T_{k',l'} \geq_B T_{k,0}$. De plus, cette propriété implique aussi que $T_{k,0} \geq_B T_{k',l'} \implies k \geq k' + l'$. Nous allons montrer la réciproque. Soient $k', l', k \geq 1$, avec $k' \geq l'$ et $k \geq k' + l'$. En reprenant les notations de la partie précédente, $S_{k,0} = 2 \cdot (\alpha^k + \beta^k)$. Or $l' \leq k - k'$ donc $S_{k',l'} \leq S_{k',k-k'}$. De plus, $S_{k,0} - S_{k',k-k'} = \alpha^k + \beta^k - \alpha^{k'}\beta^{k-k'} - \beta^{k'}\alpha^{k-k'}$. On a déjà étudié cette fonction dans la partie précédente (Étude Préliminaire), et on a montré que cette différence était positive. Ainsi, $T_{k',l'} \leq_B T_{k',k-k'} \leq_B T_{k,0}$.

Propriétés 8. *Soient $k', l', k \geq 1$. Alors $T_{k,0} \geq_B T_{k',l'} \Leftrightarrow k \geq k' + l'$. Dans les autres cas, $T_{k,0}$ et $T_{k',l'}$ sont incomparables.*

3.3.2 Avec une arité non bornée à la racine

De façon évidente, pour tous $q, m, y_1, \ldots, y_q, b_1, \ldots b_m$,
$hom(T_{y_1,y_2,\ldots y_q}, T_{b_1,b_2,\ldots b_m}) = \prod_{i=1}^{q}(b_1^{y_i} + b_2^{y_i} + \ldots b_m^{y_i})$. Soient deux arbres $T = T_{x_1,x_2,\ldots x_p}$ et $T' = T_{x'_1,x'_2,\ldots x'_n}$. On suppose que $p \leq n$. Par la propriété 5, soit $T \leq_B T'$, soit T et T' sont incomparables. Toujours par la propriété 5, $T \leq_B T' \implies x_1 + x_2 + \ldots + x_p \leq x'_1 + x'_2 + \ldots + x'_n$. On fera donc cette hypothèse. En utilisant la propriété 6, on montre que
$T \leq T'$ si et seulement si $\forall m \cdot \forall a_1 \ldots a_n \cdot \prod_{i=1}^{p}(a_1^{x_i} + a_2^{x_i} + \ldots a_m^{x_i}) \leq \prod_{i=1}^{n}(a_1^{x'_i} + a_2^{x'_i} + \ldots a_m^{x'_i})$.
On a ainsi obtenu une condition similaire à celle que voulait proposer[BH97] dans le cadre des graphes. Ceci nous permet de généraliser la caractérisation obtenue précédemment lorsque l'arité de la racine était fixée à deux.

Théorème 3. $\left(\forall k \in [1..p] \cdot \sum_{i=1}^{k} x_i \leq \sum_{j=1}^{k+n-p} x'_j\right) \Leftrightarrow T \leq_B T'$

Démonstration. La preuve détaillée étant relativement complexe, nous n'en donnons ici que les grandes lignes [3]. L'idée est de généraliser la preuve pour l'arité 2 : pour montrer que la condition est nécessaire, on considère un graphe X de la forme $T_{\alpha,1,1,1\ldots1}$ avec m noeuds à profondeur 1 qui n'ont qu'un fils. On montre que pour tout k, il existe certaines valeurs de α et de m pour lesquelles le nombre d'homomorphismes de T (respectivement T') dans X dépend étroitement de $\sum_{i=1}^{k} x_i$ (respectivement $\sum_{j=1}^{k+n-p} x'_j$).

Pour ce qui est de la réciproque, l'idée générale est que, à somme des x_i égale, c'est l'arbre dont les x_i les plus grands sont les plus petits qui est inférieur. Une fois ceci prouvé, on transforme ainsi l'arbre T' en réduisant l'arité des noeuds de forte arité x'_i et ajoutant le nombre correspondant de feuilles aux noeuds de faible arité x'_i. On obtient ainsi un arbre T" inférieur à T', plus facile à comparer avec T. On déduit $T'_B \geq T"_B \geq T_B$ □

Nous n'avons pas encore réussi à obtenir une caractérisation quand la profondeur est quelconque. Il semble néanmoins que la caractérisation devienne plus combinatoire, si tant est que l'inclusion soit décidable.

De manière plutôt surprenante, l'équivalence de requêtes est, elle, beaucoup plus simple en sémantique multiensembliste, comme nous le montrons par la suite.

4 Conjunctive Query Equivalence en sémantique multiensembliste

Nous allons ici utiliser l'interprétation en termes de graphes du problème d'équivalence de requêtes. Par définition, deux requêtes Q et Q' sont équivalentes ssi $Q \leq_B Q'$ et $Q' \leq_B Q$. En termes d'homomorphismes, si on note $G[Q]$ et $G[Q']$ les graphes associés aux requêtes Q et Q' comme expliqué plus haut[4], cela signifie que pour tout graphe X, le nombre d'homomorphismes de $G[Q]$ vers X respectant les inégalités N est le même que celui de $G[Q']$ vers X respectant les inégalités N' puisque $Q \leq_B Q' \Leftrightarrow \forall X, hom_N(G[Q], X) \leq hom_{N'}(G[Q'], X)$ et $Q' \leq_B Q \Leftrightarrow \forall X, hom_N(G[Q], X) \geq hom_{N'}(G[Q'], X)$

Cependant cette caractérisation n'est pas très pratique puisque la quantification porte sur un nombre infini de graphes X. Elle ne permet donc pas de décider si deux requêtes sont équivalentes. En revanche on va montrer que l'on peut la transformer en une caractérisation équivalente plus utile.

4.1 Caractérisation de l'équivalence en termes de surjections

Tout d'abord, si on considère deux graphes G et X, on constate que tout homomorphisme ϕ de G vers X peut s'écrire comme la composition d'une surjection σ de G vers un graphe F isomorphe à $\phi(G)$ suivie d'une injection $\tau : F \mapsto X$. Il nous faut rendre cette décomposition unique si on veut pouvoir s'en servir à des fins de dénombrement. On définit donc $inj'(F, X)$ comme le nombre d'injections de F vers X distinctes à automorphisme de F près. Plus formellement, on définit la relation d'équivalence $\mathcal{R}$ sur les morphismes injectifs $\tau : F \to X$ par : $\tau \mathcal{R} \tau'$ si et seulement si il existe un automorphisme $\rho : F \to F$ tel que $\tau' = \tau \circ \rho\}$. On note

[3] cf preuve complète en annexe

[4] voir en p.6

$[\tau]$ la classe d'équivalence d'un morphisme injectif τ pour cette relation. $inj'(F, X)$ compte le nombre de telles classes. On se donne pour tous graphes F et X un ordre arbitraire sur les injections de F dans X, de façon à définir un élément minimal pour chaque classe de $\mathcal{R}$.

Etant donné un schéma $\mathcal{S}$ et un entier n, on pose $\mathcal{G}_n$ l'ensemble des graphes à moins de n sommets sur le schéma $\mathcal{S}$ dans lequel on garde un graphe unique par classe d'isomorphisme. Soient G, X des hypergraphes définis sur un schéma $\mathcal{S}$, tels que G a au plus n sommets. Soit $\mathcal{N}$ un ensemble de contraintes d'inégalités sur G. On définit enfin $\mathcal{E}$ l'ensemble des triplets (F, τ, σ) tels que $F \in \mathcal{G}_n$, τ homomorphisme injectif de F vers X minimal dans sa $\mathcal{R}$-classe, et σ surjection de G dans F respectant les inégalités de $\mathcal{N}$ (F est l'image par σ de G).

Lemme 1. *Pour tout morphisme ϕ de G dans X, il existe un unique triplet $(F, \tau, \sigma) \in \mathcal{E}$ tel que $\phi = \tau \circ \sigma$.*

De ce lemme on déduit qu'il existe une injection de l'ensemble des homomorphismes de G dans X vers $\mathcal{E}$, donc que le nombre d'homomorphismes de G dans X respectant les inégalités de $\mathcal{N}$ est supérieur au nombre de triplets (F, τ, σ) dans $\mathcal{E}$. Comme réciproquement on peut associer à tout triplet de $(F, \tau, \sigma) \in \mathcal{E}$ le morphisme $\phi = \tau \circ \sigma$, le nombre d'homomorphismes de G dans X respectant les inégalités de $\mathcal{N}$ est égal au nombre de triplets de $\mathcal{E}$. Par suite,

Lemme 2. *Soient G, X des hypergraphes définis sur un schéma $\mathcal{S}$, tels que G a au plus n sommets. Soit $\mathcal{N}$ un ensemble de contraintes d'inégalités sur G. Alors*

$$hom_N(G, X) = \sum_{F \in \mathcal{G}_n} sur_N(G, F).inj'(F, X)$$

Théorème 4. *Soient G_1, G_2 des hypergraphes à moins de n sommets, définis sur un schéma $\mathcal{S}$. Soient $\mathcal{N}_1$ et $\mathcal{N}_2$ des ensembles de contraintes d'inégalités sur G_1 et G_2. Alors*

$$\begin{aligned} & \forall \quad X \cdot hom_{N_1}(G_1, X) = hom_{N_2}(G_2, X) \\ \Leftrightarrow \quad & \forall \quad F \in \mathcal{G}_n \cdot surj_{N_1}(G_1, F) = surj_{N_2}(G_2, F) \end{aligned}$$

De cette caractérisation on déduit un résultat important, déjà prouvé dans par exemple [NSS98] :

Corollaire. *Le problème CQE_B avec des inégalités est dans $PSPACE$.*

Démonstration. En effet il suffit d'énumérer dans une boucle tous les graphes X de taille plus petite que celle des graphes d'entrée G_1 et G_2, et de vérifier que le nombre de surjections dans ces graphes est le même pour G_1 et G_2. □

4.2 Un algorithme de décomposition pour éliminer les inégalités

Cette caractérisation de l'équivalence, proposée par E. Vee [Vee] est pratique d'un point de vue théorique, mais elle impose encore d'énumérer toutes les surjections.

Une équation toute simple donne une approche récursive différente pour tester l'équivalence. L'idée est que, à partir d'un graphe G avec un ensemble d'inégalités $\mathcal{N}$, si on prend un couple de sommets (v, v') qui n'est pas dans $\mathcal{N}$, on peut partitionner les homomorphismes de G dans un graphe X respectant $\mathcal{N}$ en deux groupes : les homomorphismes qui envoient v et v' sur le même sommet et les nombre d'homomorphismes qui les séparent : le nombre d'homomorphismes qui envoient v et v' sur des sommets disjoints est $hom_{N \cup (v,v')}(G, X)$, et le nombre d'homomorphismes qui les envoie sur le même sommet est $hom_{\mathcal{N}_{v=v'}}(G_{v=v'}, X)$, où

$\mathcal{N}_{v=v'}$ est l'ensemble d'inégalités $\mathcal{N}$ dans lequel on a renommé v' en v, et $G_{v=v'}$ le graphe G dans lequel on a contracté v et v', en nommant v le sommet ainsi obtenu. Plus formellement, $G_{v=v'}$ est donné par l'ensemble de sommets $V_G \backslash \{v'\}$, et les relations de G dans lesquelles on remplace v' par v. Ceci donne l'équation (2). En retournant cette équation, et en notant qu'on peut restreindre la partition aux homomorphismes surjectifs, on obtient :

$$\begin{aligned} hom_{\mathcal{N}}(G,X) &= hom_{\mathcal{N}\cup(v,v')}(G,X) + hom_{\mathcal{N}_{v=v'}}(G_{v=v'},X) && (2)\\ hom_{\mathcal{N}}(G,X) &= hom_{\mathcal{N}\setminus(v,v')}(G,X) - hom_{\mathcal{N}_{v=v'}}(G_{v=v'},X) && (3)\\ surj_{\mathcal{N}}(G,X) &= surj_{\mathcal{N}\setminus(v,v')}(G,X) - surj_{\mathcal{N}_{v=v'}}(G_{v=v'},X) && (4) \end{aligned}$$

On peut alors itérer l'équation (4) pour éliminer une à une les inégalités :

Algorithme 1 Algorithme pour Décomposer un graphe en éliminant les inégalités

Entrées : $(G,\mathcal{N})$ un graphe avec son ensemble d'inégalités.
Sorties : Deux ensembles de graphes sans inégalités $\mathcal{E}_G^+$ et $\mathcal{E}_G^-$ tels que

$$surj_{\mathcal{N}}(G,X) = \sum_{G_i\in\mathcal{E}_G^+} surj(G_i,X) - \sum_{G_j\in\mathcal{E}_G^-} surj(G_j,X)$$

Initialisation :
$\mathcal{E}_G^+ \leftarrow (G,\mathcal{N})$
$\mathcal{E}_G^- \leftarrow \emptyset$
Tantque il existe $(H,\mathcal{N}_H) \in (\mathcal{E}_G^+ \cup \mathcal{E}_G^-)$ tel que $\mathcal{N}_H \neq \emptyset$ **faire**
  Choisir arbitrairement $(v,v') \in \mathcal{N}_H$
  si $(H,\mathcal{N}_H) \in \mathcal{E}_G^+$ **alors**
    $\mathcal{E}_G^+ \leftarrow \mathcal{E}_G^+ \setminus (H,\mathcal{N}_H) \cup (H,\mathcal{N}_H \setminus (v,v'))$
    $\mathcal{E}_G^- \leftarrow \mathcal{E}_G^- \cup (H_{v=v'},(\mathcal{N}_H)_{v=v'})$
  sinon
    $\mathcal{E}_G^- \leftarrow \mathcal{E}_G^- \setminus (H,\mathcal{N}_H) \cup (H,\mathcal{N}_H \setminus (v,v'))$
    $\mathcal{E}_G^+ \leftarrow \mathcal{E}_G^+ \cup (H_{v=v'},(\mathcal{N}_H)_{v=v'})$
  finsi
Fin tantque
Retourner $\mathcal{E}_G^+$ et $\mathcal{E}_G^-$.

On vérifie par induction en utilisant (4) qu'à tout instant de l'algorithme on a :

$$surj_{\mathcal{N}}(G,X) = \sum_{(H,\mathcal{N}_H)\in\mathcal{E}_G^+} surj_{\mathcal{N}_H}(H,X) - \sum_{(H',\mathcal{N}_{H'})\in\mathcal{E}_G^-} surj_{\mathcal{N}_{H'}}(H',X)$$

ce qui garantit donc la correction puisque à la fin les ensembles d'inégalités sont vides. Ainsi on obtient bien à la fin de l'algorithme :

$$surj_{\mathcal{N}}(G,X) = \sum_{G_i\in\mathcal{E}_G^+} surj(G_i,X) - \sum_{G_j\in\mathcal{E}_G^-} surj(G_j,X) \quad (5)$$

(Techniquement, l'algorithme rend des ensemble de couples $(H,\emptyset)$ constitués d'un graphe et d'un ensemble d'inégalité vide, mais on assimile ces couples à leur graphe H). Une remarque

intéressante est que dans $\mathcal{E}_G^+$ on a les graphes H tel que le nombre de sommets de G moins celui de H est pair, alors que pour $\mathcal{E}_G^-$ cette différence est impaire.

Illustrons cet algorithme sur un exemple : les inégalités de $\mathcal{N}$ sont représentées en pointillé (celle de gauche en rouge, celle de droite en vert), et le graphe G en traits pleins.

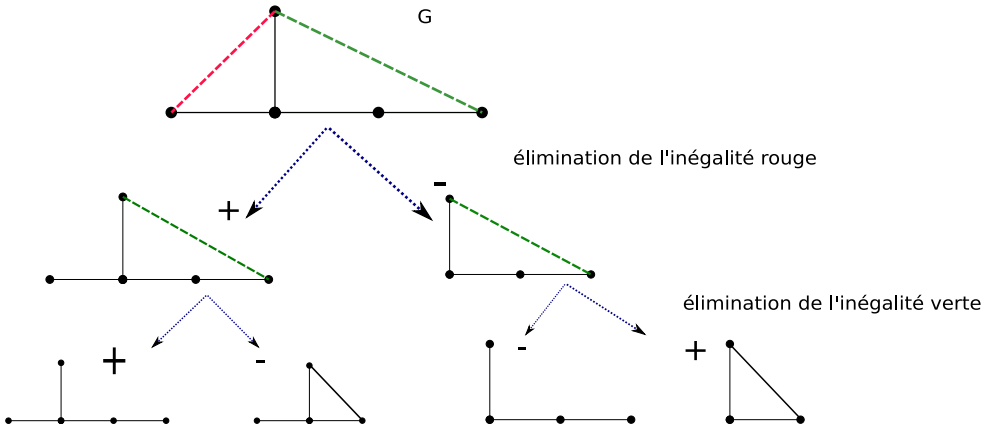

FIG. 7 – Arbre de décomposition d'un graphe G

4.3 Un algorithme pour tester l'équivalence

Soient G et G' deux requêtes avec leur ensemble d'inégalités $\mathcal{N}$ et $\mathcal{N}'$. L'idée de base est d'utiliser l'algorithme précédent(1) pour décomposer G et G'. Ceci rend des ensembles de graphes $\mathcal{E}_G^+$, $\mathcal{E}_G^-$, $\mathcal{E}_{G'}^+$ et $\mathcal{E}_{G'}^-$. Le théorème suivant permet alors de conclure.

Théorème 5.

$$G \equiv_B G' \Leftrightarrow \mathcal{E}_G^+ \simeq \mathcal{E}_{G'}^+ \text{ et } \mathcal{E}_G^- \simeq \mathcal{E}_{G'}^-$$

Démonstration. Nous montrons ce résulat en utilisant la caractérisation en termes de surjections de l'équivalence. Bien que l'idée de l'algorithme soit inédite, E. Vee propose une méthode proche[Vee] pour montrer ce résultat. On suppose préalablement que G et G' sont connectés.

$\Leftarrow$: Si $\mathcal{E}_G^+ \simeq \mathcal{E}_{G'}^+$ et $\mathcal{E}_G^- \simeq \mathcal{E}_{G'}^-$ alors comme les $G_i \in \mathcal{E}_G^+$,et les $G'_j \in \mathcal{E}_{G'}^+$ sont connectés, il y a une bijection f entre ces G_i et ces G'_j telle que $f(G_i) \simeq G'_i$. De même on obtient une bijection entre $\mathcal{E}_G^-$ et $\mathcal{E}_{G'}^-$ associant des graphes isomorphes. Donc il est évident par (5) que les requêtes G et G' sont équivalentes.

$\Rightarrow$: Supposons maintenant les requêtes équivalentes. On veut montrer que

$$\bigoplus_{H \in \mathcal{E}_G^+} H = \bigoplus_{K \in \mathcal{E}_{G'}^+} K \tag{6}$$

et

$$\bigoplus_{H' \in \mathcal{E}_G^+} H' = \bigoplus_{K' \in \mathcal{E}_{G'}^+} K' \tag{7}$$

Par (5), on a :

$$\begin{aligned} &\sum_{(H,\mathcal{N}_H)\in\mathcal{E}_G^+} surj_{\mathcal{N}_H}(H,X) - \sum_{(H',\mathcal{N}_{H'})\in\mathcal{E}_G^-} surj_{N_{H'}}(H',X) \\ = &\sum_{(K,\mathcal{N}_K)\in\mathcal{E}_{G'}^+} surj_{\mathcal{N}_K}(K,X) - \sum_{(K',\mathcal{N}_{K'})\in\mathcal{E}_{G'}^-} surj_{\mathcal{N}_{K'}}(K',X) \end{aligned} \tag{8}$$

Il est clair qu'il existe un unique H et un unique K avec n sommets, isomorphes à G et G' respectivement. On peut donc les ôter de chaque côté de (6) et de (8). Supposons enlevés ainsi de chaque côté tous les H, K, H', K' avec plus de x sommets et montrons qu'on peut restreindre les sommes aux H, K, H', K' avec moins de $x-1$ sommets. Prenons un élément maximal en nombre de sommets puis d'arêtes parmi les H, H', disons H_0. Alors $surj(H_0, H_0) > 0$, donc il existe un K_0 tel que $surj(K_0, H_0) > 0$, et le signe associé à H_0 sera le même que celui de K_0. Et on peut appliquer le même raisonnement à K_0, donc par maximalité, la surjection est aussi une injection, et $H_0 \simeq K_0$. On peut donc enlever ces graphes de chaque côté de (6) ou (7) selon le signe associé, ainsi que de (8). Ceci clôt la récurrence. On obtient ainsi (6) et (7), par récurrence sur le nombre de graphes dans la somme. □

Exemple 3. *Reprenons l'exemple 7. Posons G' le graphe ci-dessous, avec deux inégalités représentées en pointillé (celle de dessus en bleu, celle de dessous en orange). On construit un arbre de décomposition de G'. Comme les feuilles de même signe dans les arbres de décomposition pour G' et de G sont isomorphes, $G \simeq G'$.*

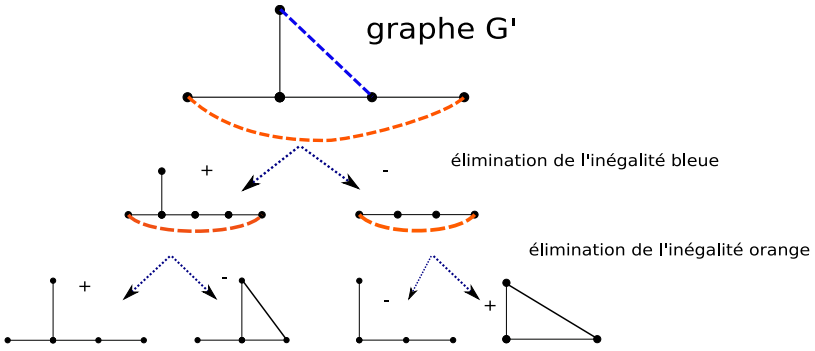

FIG. 8 – arbre de décomposition de G'

Complexité de l'algorithme récursif par décomposition : Regardons maintenant la complexité d'un algorithme basé sur cette décomposition. Tout d'abord, on observe que cet algorithme termine après $|\mathcal{N}|$ étapes de décomposition, c'est-à-dire que l'arbre de décomposition a une profondeur au plus $|\mathcal{N}|$. En effet, $|\mathcal{N}\backslash(v,v')| = |\mathcal{N}| - 1$ et $|\mathcal{N}_{v=v'}| \leq |\mathcal{N}| - 1$.

Ceci garantit un algorithme en $PSPACE$ pourvu qu'on parcoure les arbres de décomposition intelligemment. Pour commencer on ne construit pas tout l'arbre mais on le calcule "à la volée". Ensuite, on suppose fixé un ordre arbitraire sur les opérations de décomposition (c-à-d. sur le choix de H, puis d'une arête dans N_H dans l'algorithme 1) pour s'assurer de toujours obtenir le même arbre. Enfin, on parcours les arbres en ordre préfixe, classiquement. Ce sont les feuilles de l'arbre qui nous intéressent de toute façon. Pour chaque feuille (correspondant à un graphe G_i) dans l'arbre de décomposition pour G :

1. On regarde si une feuille déjà traitée (donc plus "à gauche" dans l'arbre) est isomorphe à G_i. Si oui, on passe à la feuille suivante et recommence.
2. On compte dans l'arbre le nombre de feuilles G_j isomorphes à G_i, avec leur signe.
3. On vérifie que dans l'arbre de décomposition de G', le nombre de feuilles isomorphes à G_i est le même. Sinon, les deux requêtes ne sont pas équivalentes.
4. on passe à la feuille suivante et recommence.

On retrouve bien un algorithme dans $PSPACE$ pour tester CQE_B avec des inégalités

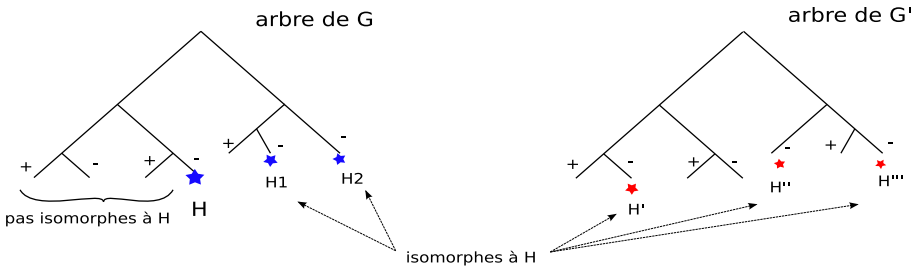

FIG. 9 – Algorithme PSPACE utilisant les arbres de décomposition

4.4 Complexité du problème CQE_B

Les résultats précédents montrent que CQE_B est dans $PSPACE$ en présence d'inégalités. La complexité exacte de CQE_B n'est toujours pas connue et nous n'avons pas réussi à la déterminer dans le cas général.

En revanche certaines bornes inférieures sont connues. En effet,

Propriétés 9.

1. $(G,\mathcal{N}) \equiv_B (G',\mathcal{N}') \implies G \equiv G'$
2. $(G,\mathcal{N}) \equiv_B (G',\mathcal{N}') \implies \mathcal{N}$ *et* $\mathcal{N}'$ *sont chromatiquement équivalents*

La première affirmation est triviale une fois que l'on a la caractérisation par les surjections : si $(G,\mathcal{N}) \equiv_B (G',\mathcal{N}')$ alors $\forall X \cdot surj_{\mathcal{N}}(G,X) = surj_{\mathcal{N}'}(G',X)$, ce qui implique en particulier pour $X = G'$ que

$$surj_{\mathcal{N}}(G,G') = surj_{\mathcal{N}'}(G',G') > 0$$

De même, $surj_{\mathcal{N}}(G',G) > 0$ donc G et G' sont isomorphes.

La deuxième affirmation mérite plus de précisions.

Définition. *Deux graphes sans boucles non orientés G et G' sont* chromatiquement équivalents *si pour tout n, ils ont le même nombre de n-coloriages. Ce qui revient à*

$$\forall n \cdot, hom(G,K_n) = hom(G',K_n)$$

où K_n est la clique à n sommets.

La deuxième affirmation de propriété 9 vient du point suivant : si X est une l-clique avec toutes les boucles présentes, que l'on notera K'_l, alors les homomorphismes de G dans X qui respectent les inégalités de $\mathcal{N}$ sont exactement les homomorphismes de $\mathcal{N}$ dans la l-clique K_l (sans boucles). En effet, les arêtes de G n'imposent plus la moindre contrainte.

Par suite,

$$\begin{array}{rcl}(G,\mathcal{N}) \equiv_B (G',\mathcal{N}') & \Longrightarrow & \forall K'_l \cdot hom_{\mathcal{N}}(G,K'_l) = hom_{\mathcal{N}'}(G',K'_l) \\ & \Longrightarrow & \forall K_l \cdot hom(\mathcal{N},K_l) = hom(\mathcal{N}',K_l)\end{array}$$

d'où l'affirmation.

Corollaire.

1. *CQE_B avec des inégalités est plus difficile que l'isomorphisme de Graphes*
2. *CQE_B avec des inégalités est plus difficile que l'équivalence chromatique*

Démonstration. Si les ensembles d'inégalités $\mathcal{N}$ et $\mathcal{N}'$ sont vides, alors par la première affirmation dans Propriétés 9, $(G,\emptyset) \equiv_B (G',\emptyset)$ si et seulement si G et G' sont isomorphes.

Si maintenant on prend pour G et G' deux ensembles de sommets isolés, la seconde affirmation de Propriétés 9 implique que $(G,\mathcal{N}) \equiv_B (G',\mathcal{N}')$ si et seulement si $\mathcal{N}$ et $\mathcal{N}'$ sont chromatiquement équivalents. □

4.5 Avec peu ou beaucoup d'inégalités, CQE se ramène à l'isomorphisme de graphe

Théorème 6. *Lorsque le nombre d'inégalités est logarithmique en la taille du graphe (c-à-d. quand $|\mathcal{N}| \leq log(|G|)$), le problème CQE_B est équivalent à l'isomorphisme de graphe.*

Démonstration. En effet, on a déjà montré que l'isomorphisme se réduisait à CQE_B sans inégalités donc a fortiori à CQE_B avec un nombre au plus logarithmique d'inégalités.

Et pour l'autre sens, on reprend l'algorithme de décomposition : si le cardinal de $\mathcal{N}$ est de taille logarithmique, alors l'arbre de décomposition est de taille polynomiale, donc on peut calculer ses feuilles en temps polynomial, et il ne reste plus qu'à tester que $\mathcal{E}_G^+ \bigoplus \mathcal{E}_G^- \simeq \mathcal{E}_{G'}^+ \bigoplus \mathcal{E}_{G'}^-$. Ceci fournit la réduction vers l'isomorphisme de graphe. □

Théorème 7. *Lorsque le nombre de couples de sommets non reliés par des inégalités est logarithmique en la taille du graphe (c-à-d. quand $\frac{1}{2} \cdot |V_G|^2 - |\mathcal{N}| \leq log(|G|)$), le problème CQE_B est équivalent à l'isomorphisme de graphe.*

Démonstration. Pour cela on utilise la formule(2). Cela donne un algorithme de décomposition, et on montre que les requêtes sont équivalentes si et seulement si les feuilles obtenues dans l'arbre de décomposition de G sont isomorphes à celles obtenues pour G' □

En effet si on applique récursivement la décomposition :

$$\forall X \cdot surj_N(G,X) = surj_{N \cup (v,v')}(G,X) + surj_{N_{v=v'}}(G_{v=v'},X)$$

on obtient à la fin de l'algorithme un ensemble de graphes G_i avec leur ensemble d'inégalités $N_i = V_{G_i}^2$, tels que

$$\forall X \cdot surj_N(G,X) = \sum_i surj_{N_i}(G_i,X) \tag{9}$$

Cet ensemble $\{G_i|i\}$ dépend a priori de quelle arête on choisit de rajouter à chaque étape de l'algorithme. On considère pour la suite un ensemble correspondant à un choix arbitraire. Nous allons montrer que les requêtes G et G' sont équivalentes ssi

$$\bigoplus_i G_i \cong \bigoplus_j G'_j \tag{10}$$

On suppose préalablement que G et G' sont connectés.

$\Leftarrow$: Si $\bigoplus_i G_i \cong \bigoplus_j G'_j$, alors comme les G_i, G'_j sont connectés, il y a une bijection f entre les G_i et les G'_j telle que $G_i \cong G'_{f(i)}$. Donc il est évident par (9) que les requêtes G et G' sont équivalentes.

$\Rightarrow$: Supposons maintenant les requêtes équivalentes. On a donc

$$\sum_i surj_{N_i}(G_i, X) = \sum_j surj_{N'_j}(G'_j, X) \tag{11}$$

Il est clair qu'il existe un unique G_i et un unique G'_j avec n sommets, isomorphes à G et G' respectivement. On peut donc les ôter de chaque côté de (10) et de (11). Supposons enlevés ainsi de chaque côté tous les G_i, G'_j avec plus de x sommets et montrons qu'on peut restreindre les sommes aux G_i, G'_j avec moins de $x-1$ sommets. Prenons un élément maximal en nombre de sommets parmi les G_i, disons, G_k. Alors $surj_{N_k}(G_k, G_k) > 0$, donc il existe un G'_l tel que $surj_{N'_l}(G'_l, G_k) > 0$. Or N'_l contient toutes les paires de sommets de G'_l, donc la surjection est aussi une injection, et $G'_l \cong G_k$. On obtient ainsi (10), par récurrence sur le nombre de graphes dans la somme.

Discussions

En représentant les requêtes sous forme de graphes, nous avons donc réussi à donner un algorithme pour tester l'équivalence de requêtes conjonctives en sémantique multiensembliste. Cet algorithme travaille en espace polynomial dans le cas général, mais sa complexité est celle de l'isomorphisme de graphe pour une large classe de requêtes : toutes les requêtes contenant soit beaucoup soit peu d'inégalités. En revanche, il pourrait être intéressant de rechercher une classe de requêtes avec inégalités pour lesquelles la complexité de l'équivalence est faible. Mais une telle condition devra probablement porter à la fois sur les prédicats et sur les inégalités.

C'est avec cette même représentation que nous avons étudié l'inclusion de requêtes (sans inégalités, bien sûr). Nous avons prouvé plusieurs propriétés intéressantes lorsque les requêtes Q et Q' sont des arbres. Par exemple, il faut que la hauteur h des deux arbres soit la même pour qu'on ait $Q \leq_B Q'$. De plus, il suffit de regarder le résultat de l'évaluation de ces requêtes sur des arbres de profondeur h. Enfin, nous avons montré sur le cas des arbres à profondeur 2 que le problème devient vite très combinatoire, et qu'il impose de pouvoir comparer certaines formes de polynômes, donc il y a un espoir de pouvoir réduire le 10e problème de Hilbert à l'inclusion de requêtes. La complexité de l'inclusion de requêtes reste donc un problème ouvert.

Références

[And98] A. Andrzejak. An algorithm for the tutte polynomial of graphs of bounded treewidth. *Discrete Mathematics*, 190 :39–54, 1998.

[BH97] N. Brisaboa and H. Hernández. Testing bag-containment of conjunctive queries. *Acta Informatica*, 34(7) :557–578, 1997.

[CM93] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the 12th ACM Symposium on the Principles of Database Systems*, 1993.

[CV93] S. Chaudhuri and M. Vardi. Optimization of *Real* conjunctive queries. In *Proceedings of the 12th ACM Symposium on the Principles of Database Systems*, 1993.

[DG00] M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17 :260–289, 2000.

[IR95] Y. Ioannidis and R. Ramakrishnan. Containment of conjunctive queries : beyond relations as sets. *ACM Transactions on Database Systems*, 20(3) :288–324, September 1995.

[JKV06] T.S. Jayram, Phokion G. Kolaitis, and Erik Vee. The containment problem for *Real* conjunctive queries with inequalities. *PODS'06*, June 2006.

[MRAG06] J.A. Makowsky, Udi Rotics, Ilya Averbouch, and Benny Godlin. Computing graph polynomials on graphs of bounded clique-width. *Graph-Theoretics Concepts in Computer Science*, 2006.

[Nob98] S.D. Noble. Evaluating the tutte polynomial for graphs of bounded treewidth. *Combinatorics, Probability and Computing*, 7 :307–321, 1998.

[NSS98] W. Nutt, Y. Sagiv, and S. Shurin. Deciding quivalences among aggregate queries. In *17th ACM Symposium on the Principles of Database Systems*, 1998.

[SY80] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *JACM*, 27 :663–655, 1980.

[vdM97] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *Journal of Computer and System Sciences*, 54(1) :113–135, 1997.

[Vee] E. Vee. personal communication.

[Vee06] E. Vee. A note on 'testing bag-containment of conjunctive queries'. *Acta Informatica*, submitted in 2006.

5 ANNEXE A : L'équivalence chromatique

Définition. *Deux graphes sans boucles non orientés G et G' sont* chromatiquement équivalents *si pour tout n, ils ont le même nombre de n-coloriages. Ce qui revient à*

$$\forall n\cdot, hom(G, K_n) = hom(G', K_n)$$

où K_n est la clique à n sommets.

Propriétés 10. *Si deux graphes sans boucles et non orientés N et N' sont chromatiquement équivalents alors*

- *N et N' ont le même nombre de sommets*
- *N et N' ont le même nombre d'arêtes*
- *Soit l la taille du plus petit cycle dans N. Alors le plus petit cycle de N' est aussi de taille l et N et N' ont le même nombre de plus petits cycles.*
- *on connaît toute une série de graphes G tels que seuls les graphes isomorphes à G lui sont chromatiquement équivalents : c'est le cas des cliques, des graphes en forme de θ...*

L'équivalence chromatique est un problème ancien et très étudié, dont la complexité exacte reste ouverte. Néanmoins, la fonction f_G qui à tout k associe le nombre de k coloriages de G est un polynôme en k, appelé *polynôme chromatique* de G, et on sait que calculer le polynôme chromatique d'un graphe est un problème $\#P - complet$ [DG00]. Ce qui ne donne toutefois pas de borne inférieure sur l'équivalence chromatique : on n'a pas forcément besoin de calculer f_G et $f_{G'}$ pour décider si $f_G = f_{G'}$. On connaît en outre des classes de graphes sur lesquelles la complexité de l'équivalence chromatique est polynomiale : calculer le polynôme chromatique se fait en temps polynomial sur des graphes de clique-width bornée [MRAG06], et sur les graphes de tree-width bornée [And98] [Nob98].

6 ANNEXE B : $CQCon_B$ est indécidable en présence d'inégalités, ou pour les unions de requêtes

Commençon par définir formellement l'évaluation d'une union de requêtes conjonctives.

Définition. *Une union de requêtes conjonctives est la donnée de $n \geq 1$ et d'un ensemble de m requêtes conjonctives $Q_1, \ldots, Q_m$ avec le même n-uplet de variables liées $\vec{x}$. Son évaluation sur une base de données en sémantique multiensembliste est un multiensemble de n-uplets :*

$$Eval_B(\bigcup_{i=0}^{m} Q_i)(c_1, \ldots, c_n) = \sum_{i=0}^{m} Eval_B(Q_i)(c_1, \ldots, c_n)$$

Le problème que l'on va réduire a l'inclusion d'union de requêtes est dérivé du 10 e problème de Hilbert :

Lemme 3. *Le problème :*
Donnée : deux polynômes $\phi(x_1, \ldots, x_k)$ et $\psi(x_1, \ldots, x_k)$ à coefficients entiers positifs et sans terme constant,
Question : a-t-on $\phi(x_1, \ldots, x_k) \leq \psi(x_1, \ldots, x_k)$ pour tous $x_1, \ldots, x_k \geq 0$?
est indécidable.

Etant donnée une instance de ce problème nous construisons deux unions de requêtes conjonctives $P_1 = \bigcup_{i=0}^{m} Q_i$ et $P_2 = \bigcup_{j=0}^{m'} Q'_j$ telles que $\bigcup_{i=0}^{m} Q_i \leq_B \bigcup_{j=0}^{m'} Q'_j$ ssi $\phi(x_1, \ldots, x_k) \leq \psi(x_1, \ldots, x_k)$ pour tous $x_1, \ldots, x_k \geq 0$.

Décomposons ϕ en somme de monômes : $\phi(x_1, \ldots, x_k) = \sum_{i=1}^{s} a_i \cdot x_1^{c_{i,1}} x_2^{c_{i,2}} \cdots x_k^{c_{i,k}}$ avec pour tout i, a_i un entier positif, et pour tout j, $c_{i,j}$ un entier positif ou nul tel que pour tout i,au moins un des $c_{i,j}$ est non nul.

Pour tout $i \in [1..m]$, on construit a_i requêtes de la forme

$$\begin{aligned} Q_i() :- \quad & X_1(v_{1,1}), X_1(v_{1,2}), \ldots, X_1(v_{1,c_{i,1}}), \\ & X_2(v_{2,1}), X_2(v_{2,2}), \ldots, X_2(v_{2,c_{i,2}}), \\ & X_n(v_{n,1}), X_n(v_{n,2}), \ldots, X_n(v_{n,c_{i,n}}) \end{aligned}$$

La requête (sans variable libre) P_1 est l'union de ces $(\sum_{i=1}^{s} a_i)$ requêtes.
On décompose de façon similaire le polynôme ψ pour construire la requête P_2.

Lemme 4. *$\phi(x_1, \ldots, x_k) \leq \psi(x_1, \ldots, x_k)$ ssi $P_1 \leq_B P_2$.*

Démonstration. $\Rightarrow$: Soit D une base de données arbitraire. Pour tout i, notons $|X_i|_D$ le nombre d'éléments $X_i(v)$ dans D. $Eval(P_1, D)() = \phi(|X_1|_D, |X_2|_D, \ldots, |X_n|_D)$. Donc $\forall x_1 \ldots x_n \cdot \phi(x_1, \ldots, x_k) \leq \psi(x_1, \ldots, x_k) \implies P_1 \leq_B P_2$. $\Leftarrow$: Réciproquement, si $P_1 \leq_B P_2$, alors étant donné une valuation arbitraire $m_1 \cdots m_n$ pour $x_1 \cdots x_n$, on construit une base D qui contient pour tout i m_i faits X_i :
en d'autres termes, soient $c_{1,1}, c_{1,2}, ...c_{n,m_n}$ des constantes distinctes. On pose
$D = \{X_1(c_{1,1}), \ldots X_1(c_{1,m_1}), X_2(c_{2,1}), \ldots, X_2(c_{2,m_2}), \ldots X_n(c_{n,m_n})\}$.
Alors $Eval(P_1, D)() = \phi(|X_1|_D, |X_2|_D, \ldots, |X_n|_D) = \phi(m_1, m_2, \ldots, m_n)$ donc $\phi(m_1, m_2, \ldots, m_n) \leq \psi(m_1, m_2, \ldots, m_n)$. □

De cette preuve proposée par Ioannidis et Ramakrishnan [IR95] on déduit l'indécidabilité de l'union de requêtes conjonctives :

Théorème 8. *Le problème :*
Donnée : deux unions de requêtes conjonctives $\bigcup_{i=1}^{n} Q_i$ et $\bigcup_{j=1}^{m} Q'_j$
Question : a-t-on $\bigcup_{i=1}^{n} Q_i \leq_B \bigcup_{j=1}^{m} Q'_j$?
est indécidable.

La preuve d'indécidabilité de $CQCon_B$ avec inégalités[JKV06] est similaire, quoique beaucoup plus compliquée car il est bien plus difficile de simuler la somme avec des inégalités sur les variables qu'avec l'union de requêtes conjonctives.

7 ANNEXE C : Preuves complémentaires

Voici les preuves qui n'ont pas été incluses dans le rapport dans un souci de concision voire de clarté.

Preuve de l'exemple2

Exemple. *Considérons les requêtes :*
$Q_1(x, y) : -R(x), T(y), S(z, x), S(t, y)$ *et*
$Q_2(x, y) : -R(x), T(y), S(z, t), S(w, t)$.
On peut vérifier que $Q_1 \leq_B Q_2$.

Démonstration. Soit D une base de données sur le domaine $\mathcal{C}$. Pour tous a et b dans $\mathcal{C}$,

$$\begin{aligned} Eval(Q_1, D)(a,b) &= |R(a)|_D \cdot \left(\sum_{\tau_u, \tau_v \in \mathcal{C}} |S(\tau_u, a)|_D \cdot |S(\tau_v, b)|_D \right) \cdot |T(b)|_D \\ Eval(Q_2, D)(a,b) &= |R(a)|_D \cdot \left(\sum_{\tau_u, \tau_v, \tau_y \in \mathcal{C}} |S(\tau_u, \tau_y)|_D \cdot |S(\tau_v, \tau_y)|_D \right) \cdot |T(b)|_D \end{aligned}$$

Nous allons montrer que

$$\sum_{\tau_u, \tau_v} |S(\tau_u, a)|_D \cdot |S(\tau_v, b)|_D \leq \sum_{\tau_u, \tau_v, \tau_y} |S(\tau_u, \tau_y)|_D \cdot |S(\tau_v, \tau_y)|_D$$

Dans le cas où $a = b$, c'est évident. Sinon, on considère les vecteurs $C_1 = (|S(\tau_u, a)|_D)_{\tau_u \in \mathcal{C}}$ et $C_2 = (|S(\tau_v, b)|_D)_{\tau_v \in \mathcal{C}}$ représentant le nombre d'éléments $S(x, a)$ et $S(x, b)$ dans D pour tous les x de $\mathcal{C}$.

$$\begin{aligned} \sum_{\tau_u, \tau_v, \tau_y} |S(\tau_u, \tau_y)|_D * |S(\tau_v, \tau_y)|_D &\geq \sum_{\tau_u, \tau_v} |S(\tau_u, a)|_D \cdot |S(\tau_v, a)|_D + \\ &\quad \sum_{\tau_u, \tau_v} |S(\tau_u, b)|_D \cdot |S(\tau_v, b)|_D \\ &\geq C_1.C_1 + C_2.C_2 \\ &\geq 2.C_1.C_2 \\ &\geq C_1.C_2 \\ &\geq \sum_{\tau_u, \tau_v} |S(\tau_u, a)|_D \cdot |S(\tau_v, b)|_D \end{aligned}$$

□

Preuves des propositions au sujet de $CQCon_B$ sur les arbres

Soient deux arbres $T = T_{x_1, x_2, \ldots x_p}$ et $T' = T_{x'_1, x'_2, \ldots x'_n}$. On suppose sans perte de généralité $x_1 \geq \ldots \geq x_p \geq 1$, $x'_1 \geq \ldots \geq x'_n \geq 1$, $p \leq n$ et $x_1 + x_2 + ... + x_p \leq x'_1 + x'_2 + ... + x'_n$. En utilisant la propriété 6, on montre que
$T \leq T'$ si et seulement si $\forall m \cdot \forall a_1 \ldots a_n \cdot \prod_{i=1}^{p} (a_1^{x_i} + a_2^{x_i} + \ldots a_m^{x_i}) \leq \prod_{i=1}^{n} (a_1^{x'_i} + a_2^{x'_i} + \ldots a_m^{x'_i})$.
Ceci mène aux propriétés suivantes :

Théorème. (3) $T \leq_B T' \Leftrightarrow \forall k \in [1..p] \cdot \sum_{i=1}^{k} x_i \leq \sum_{j=1}^{k+n-p} x'_j$

L'idée de la preuve est similaire à celle présentée dans le cas d'une racine d'arité 2. Cependant, il y a ici plus de paramètres, ce qui rend la preuve moins lisible. Une étude de cas étant impossible (d'autant plus que l'unique avantage de l'étude de cas était de donner une intuition des mécanismes en jeu), on décompose plutôt la preuve en deux implications.

Propriétés 11. $T \leq_B T' \implies \forall k \in [1..p] \cdot \sum_{i=1}^{k} x_i \leq \sum_{j=1}^{k+n-p} x'_j$

Démonstration. Supposons qu'il existe k tel que $x_1 + x_2 + \dots x_k > x'_1 + x'_2 + \dots + x'_{k+n-p}$.
On va montrer qu'il n'est pas possible d'avoir $T_{x_1,x_2,\dots x_p} \leq_B T_{x'_1,x'_2,\dots x'_n}$.
Soient m et α des entiers. On prend $X = T_{\alpha,\underbrace{1,\dots,1}_{m}}$.
Alors, $hom(T,X) = (\alpha^{x_1} + m) \cdots (\alpha^{x_2} + m) \dots (\alpha^{x_n} + m)$.

$$\begin{aligned} hom(T,X) &= \alpha^{x_1+x_2+\dots x_p} \\ &+ m \cdot \left(\alpha^{x_1+\dots+x_{p-1}} + \dots\right) \\ &+ m^2 \cdot \left(\alpha^{x_1+\dots+x_{p-2}} + \dots\right) \\ &+ \dots \\ &+ m^{p-k} \cdot \left(\alpha^{x_1+\dots+x_k} + \dots\right) \\ &+ \dots \\ &+ m^{p-1} \cdot (\alpha^{x_1} + \alpha^{x_2} + \dots) \\ &+ m^p \end{aligned}$$

(On a écrit $hom(T,X)$ comme un polynôme en m. Pour chaque coefficient de ce polynôme en m, on a considéré ce coefficient comme un polynôme en α, et on n'en a écrit que le terme dominant).
De même $hom(T',X) = \sum_{i=0}^{m} m^i \cdot (\alpha^{x'_1+\dots+x'_{n-i}} +$ des termes de degré moindre).

Posons $m = \alpha^{x'_{k+n-p}}$. On a alors pour tout $l \in [k+n-p+1..n]$ $\alpha^{x'_l} \leq m$.
Donc pour tout $j \geq k+n-p$, $\prod_{i=k+n-p+1}^{j} \alpha^{x'_i} \leq m^{j-(k+n-p)}$. Par suite les termes $m^{n-j} \cdot \left(\alpha^{x'_1+x'_2+\dots x'_j} + \dots\right)$ considérés en tant que polynômes en α sont de degré inférieur (ou égal) au degré de $m^{p-k} \cdot \left(\alpha^{x'_1+x'_2+\dots x'_{k+n-p}} + \dots\right)$.
De même, pour tout $l \in [1..k+n-p]$ $\alpha^{x'_l} \geq m$. Donc pour tout $j \leq k+n-p$ les termes $m^{n-j} \cdot \left(\alpha^{x'_1+x'_2+\dots x'_j} + \dots\right)$ considérés en tant que polynômes en α sont de degré inférieur (ou égal) au degré de $m^{p-k} \cdot \left(\alpha^{x'_1+x'_2+\dots x'_{k+n-p}} + \dots\right)$.
Ceci implique que tous les termes dans $hom(T',X)$ sont de degré inférieur ou égal à
$(p-k) \cdot x'_{k+n-p} + (x'_1 + x'_2 + \dots x'_{k+n-p})$. Or par hypothèse,
$(p-k) \cdot x'_{k+n-p} + (x_1 + x_2 + \dots x_k) > (p-k) \cdot x'_{k+n-p} + (x'_1 + x'_2 + \dots x'_k)$. Ainsi, le degré en α de $hom(T,X)$ est supérieur.
D'où il existe une valeur de α pour laquelle $hom(T,X) > hom(T',X')$. On en conclut qu'on n'a pas $T \leq_B T'$. □

Propriétés 12. $\left(\forall k \in [1..p] \cdot \sum_{i=1}^{k} x_i \leq \sum_{j=1}^{k+n-p} x'_j\right) \implies T \leq_B T'$

Étude de fonction préliminaire : Soit $n \geq 1$. Soient $a_1, a_2, \dots a_n \geq 1$ n entiers, et $e_1 \dots, e_n$ n entiers ordonnés par ordre décroissant. On note $\mathcal{S}_n$ l'ensemble des permutations à n éléments. Notons $S^{a_1,\dots a_n}_{e_1,e_2,\dots,e_n}$ ou plutôt $S_{e_1,\dots,e_n}$ la somme $\sum_{\sigma \in \mathcal{S}_n} \prod_{l=1}^{n} a_l^{e_{\sigma(l)}}$.
Posons $t = \sum_{l=1}^{n} e_l$. Pour tous i,j avec $i > j$, si on maintient constants les e_k pour tout $k \notin \{i,j\}$, ainsi que la somme t, alors $S_{e_1,\dots,e_n}$ croît avec e_i. En effet, $e_j = t - \sum_{\substack{l=1 \\ l \neq i,j}}^{n} e_l - e_i$ donc

$\frac{\partial S_{e_1,\dots e_n}}{\partial e_i} = \sum_{\sigma\in\mathcal{S}_n}(\ln a_{\sigma^{-1}(i)} - \ln a_{\sigma^{-1}(j)})(\prod_{l=1}^n a_l^{e_{\sigma(l)}})$. Maintenant, si pour toute permutation σ on note σ' la permutation $\sigma' = \sigma \circ \tau_{i,j}$, c'est-à-dire la permutation qui inverse les images de i et j, alors

$$\frac{\partial S_{e_1,\dots e_n}}{\partial e_i} = \sum_{\substack{\sigma\in\mathcal{S}_n \\ \sigma(i)<\sigma(j)}} (\ln a_{\sigma^{-1}(i)} - \ln a_{\sigma^{-1}(j)})(\prod_{l=1}^n a_l^{e_{\sigma(l)}} - \prod_{l=1}^n a_l^{e_{\sigma'(l)}}).$$

Or le signe de $\ln a_{\sigma^{-1}(i)} - \ln a_{\sigma^{-1}(j)}$ est le même que celui de $(\prod_{l=1}^n a_l^{e_{\sigma(l)}} - \prod_{l=1}^n a_l^{e_{\sigma'(l)}})$. Par suite, tous les termes de la somme sont positifs, donc la dérivée est positive. Ceci permet de conclure que $S_{e_1,\dots e_n}$ croît bien avec e_i.

Application aux homomorphismes : Soit $m \geq 1$, et $a_1, a_2, \dots a_m$ m entiers. Notons $X = T_{a_1,\dots a_m}$, $T = T_{e_1\dots e_n}$, $t_1, t_2, \dots t_m$ les noeuds de X à profondeur 1 (d'arité respective e_1, e_2 ...), et $v_1, v_2, \dots v_m$ les noeuds de X à profondeur 1 (d'arité respective a_1, a_2 ...). Pour tous n et $e_1 \geq \dots e_n$, les homomorphismes de $T_{e_1\dots e_n}$ dans X peuvent être partitionnés selon l'image de $\{t_1, \dots t_n\}$: les homomorphismes qui envoient les t_i sur des sommets tous différents $\{v_{i_1}, \dots v_{i_n}\}$ avec $\forall j \neq k \cdot v_{i_j} \neq v_{i_k}$ sont au nombre de $S_{e_1,\dots e_n}^{a_{i_1},\dots a_{i_n}}$ les homomorphismes qui envoient les t_i (de façon surjective) sur $(n-1)$ sommets $\{v_{i_1}, \dots v_{i_{n-1}}\}$ sont au nombre de $\sum_{j=1}^{n-1} S_{e_1,\dots e_n}^{a_{i_1},\dots a_{i_{n-1}},a_{i_j}}$ (car en ce cas il y a un sommet v_{i_j} sur lequel sont envoyés deux sommets). On voit que de la même façon, on peut décomposer en somme de termes de la forme $S_{e_1,\dots,e_n}$ les homomorphismes qui envoient les t_i sur un ensemble quelconque de sommets $\{v_{i_1} \dots v_{i_k}\}$.

Par conséquent, $hom(T_{e_1,\dots,e_n}, X)$ est la somme de termes de la forme $S_{e_1,\dots,e_n}$. Donc pour tous i, j avec $i > j$, $hom(T_{e_1,\dots,e_n}, X)$ croît avec e_i lorsque la somme $e_1 + e_2 + \dots e_n$ est fixée et seuls e_i et e_j varient, les autres paramètres e_k étant fixés.

Conclusion : Pour conclure, nous sommes amenés à effectuer une étude de cas sur l'ordre entre les x_i et les x'_j.

Tout d'abord il convient de noter que si il existe $i \leq p$ et $j \leq n$ tels que $x_i = x'_j$ alors, comme pour tout X, $hom(T_{x_1,\dots x_p}, X) = hom(T_{x_1,\dots x_{i-1},x_{i+1},\dots x_p}, X) \cdot hom(T_{x_i}, X)$, on a $T \leq_B T'$ si et seulement si $T_{x_1,\dots x_{i-1},x_{i+1},\dots x_p} \leq_B T_{x'_1,\dots x'_{j-1},x'_{j+1},\dots x'_n}$.

On raisonne par récurrence : Soit $N \in \mathbb{N}$. On suppose la propriété prouvée lorsque $p+n \leq N$. On suppose donc que $p+n = N+1$ et on montre que l'implication reste valide. On suppose $\forall k \in [1..p] \cdot \sum_{i=1}^k x_i \leq \sum_{j=1}^{k+n-p} x'_j$.

Si $x_p \leq x'_n$. Alors $T \leq_B T_{x_1,\dots,x_{p-1},x'_n}$. De plus, $\forall k \in [1..p-1] \cdot \sum_{i=1}^k x_i \leq \sum_{j=1}^{k+n-p} x'_j$ donc par hypothèse de récurrence, $T_{x_1,\dots x_{p-1}} \leq_B T_{x'_1,\dots x'_{n-1}}$. Par la remarque ci-dessus, on en déduit $T \leq_B T_{x_1,\dots,x_{p-1},x'_n} \leq_B T'$.

Sinon $x_p > x'_n$. Alors on va utiliser l'étude préliminaire pour montrer que $T \leq_B T'$. Soit β le plus petit entier tel que $\sum_{i=n-\beta}^{n-1}(x'_i - 1) \geq x_p - x'_n$. Construisons la série d'arbres suivante : $T^0 = T'$, et pour tout $i \in [1..\beta-1]$, on note $T_{y_1,\dots,y_n}$ l'arbre T^{i-1}, puis pour obtenir T^i, on augmente sa $n^{\text{ème}}$ composante de $y_{n-i} - 1$, et on ôte $y_{n-i} - 1$ à sa $(n-i)^{\text{ème}}$ composante : $T^i = T_{y_1,\dots,y_{n-i-1},1,y_{n-i+1},\dots,(y_n+y_{n-i}-1)}$. Enfin, notons $T_{y_1,\dots,y_n}$ l'arbre $T^{\beta-1}$, on pose $\alpha = x_p - y_n$, et on enlève α à sa $\beta^{\text{ème}}$ composante et l'ajoute à y_n : on obtient ainsi l'arbre $T^\beta = T_{y_1,\dots,y_{n-\beta-1},y_{n-\beta}-\alpha,\dots,y_n+\alpha}$.

Par l'étude précédente, il est clair que $T^0 \geq_B T^1 \geq_B \dots T^\beta$. De plus, si on note $T_{z_1,\dots z_n}$ l'arbre T^β (avec donc en dernière composante x_p, précédé d'une série de composante représen-

tant une arité un), il est clair que $\forall k \in [1..p] \cdot \sum_{i=1}^{k} x_i \leq \sum_{j=1}^{k+n-p} z_j$. Ainsi, comme $x_p = z_n$, la remarque liminaire et l'hypothèse de récurrence permettent de conclure : $T \leq T^\beta$ donc $T \leq_B T'$.

Il reste à traiter le cas de base de la récurrence. Il suffit de remarquer que lorsque T est d'arité un à la racine, l'équivalence est trivialement vraie.

Table des matières

1 Les Requêtes Conjonctives **2**
1.1 Définitions 2
1.2 Sémantiques 3
1.2.1 Sémantique ensembliste 3
1.2.2 Sémantique multiensembliste 4
1.3 Des requêtes conjonctives aux homomorphismes de graphes 5

2 Les résultats en sémantique ensembliste **7**

3 Conjunctive Query Containment en sémantique multiensembliste **7**
3.1 Généralités, propriétés élémentaires sur les arbres 8
3.2 Vers la décidabilité ? 10
3.3 Propriétés plus combinatoires 11
3.3.1 Caractérisation pour les arbres de profondeur 2 et d'arité 2 à la racine . 11
3.3.2 Avec une arité non bornée à la racine 13

4 Conjunctive Query Equivalence en sémantique multiensembliste **14**
4.1 Caractérisation de l'équivalence en termes de surjections 14
4.2 Un algorithme de décomposition pour éliminer les inégalités 15
4.3 Un algorithme pour tester l'équivalence 17
4.4 Complexité du problème CQE_B 19
4.5 Avec peu ou beaucoup d'inégalités, CQE se ramène à l'isomorphisme de graphe 20

Références **22**

5 ANNEXE A : L'équivalence chromatique **23**

6 ANNEXE B : $CQCon_B$ est indécidable en présence d'inégalités, ou pour les unions de requêtes **23**

7 ANNEXE C : Preuves complémentaires **24**