

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ MONTPELLIER II
— SCIENCES ET TECHNIQUES DU LANGUEDOC —

MÉMOIRE DE STAGE DE MASTER

SPÉCIALITÉ : **Recherche en Informatique**
Mention : **Informatique, Mathématiques, Statistiques**

effectué au laboratoire LIRMM/INFO

—
sous la direction de JEAN-FRANÇOIS BAGET, MARIE-LAURE MUGNIER

Graphes conceptuels et bases de données
Utilisation de clés au cours de la projection

par

Daniel Luedemann

Soutenu le 4 septembre 2006

Table des matières

1	Exposé du problème	4
1.1	Graphes conceptuels	4
1.1.1	Syntaxe	4
1.1.2	Sémantique	5
1.1.3	Homomorphisme	6
1.2	Bases de données	7
1.2.1	Syntaxe	7
1.2.2	Sémantique	9
1.2.3	Réponse à une requête	10
1.2.4	Notations	10
2	Transformations et équivalences	11
2.1	Bases de données avec support	11
2.1.1	Syntaxe	11
2.1.2	Réponse à une requête	11
2.2	Équivalence entre GC et BD + \mathcal{S}	12
2.2.1	Transformations	12
2.2.2	Théorème	13
2.3	Élimination du support d'un GC	15
2.3.1	Réification	15
2.3.2	Aplatissement et expansion	19
2.4	Équivalence entre BD + \mathcal{S} et BD	24
3	Résultats expérimentaux	26
3.1	Générateur de graphes	26
3.2	Tests	28
3.3	Tests - deuxième essai	31
3.3.1	Exemple	31
3.3.2	Résultats	33
3.4	Conclusion	34

A Influence des transformations dans le temps de projection 35

Chapitre 1

Exposé du problème

1.1 Graphes conceptuels

La théorie des graphes conceptuels (GC) [Sow76] est un formalisme général de représentation des connaissances. Les graphes conceptuels permettent de représenter les connaissances de manière précise et agréable à lire, du moins pour les graphes de petite taille.

1.1.1 Syntaxe

Support

Un support définit le vocabulaire de base avec lequel on représente des connaissances sur un domaine. Ce support est défini de la manière suivante :

Définition 1 (Support) *Un support \mathcal{S} est un triplet :*

$$\mathcal{S} = (T_C, T_R, \mathcal{M})$$

- T_C, T_R et \mathcal{M} sont des ensembles deux à deux disjoints ;
- T_C , l'ensemble des types de concept, est muni de la relation d'ordre \leq_C et possède un plus grand élément, le type universel noté \top ;
- T_R , l'ensemble des types de relation, est partitionné en ensembles de types de relation de même arité : $T_R = T_R^1 \cup T_R^2 \cup \dots \cup T_R^k$, où T_R^i est l'ensemble des types de relation d'arité i , muni de la relation d'ordre \leq_i , et possède un plus grand élément noté \top_i ;
- $\mathcal{M} = \mathcal{I} \cup \{*\}$ est l'ensemble des marqueurs, composé de l'ensemble des marqueurs individuels \mathcal{I} ainsi que le marqueur générique $*$. Dans \mathcal{M} , $*$ est le plus grand élément et les éléments de \mathcal{I} sont deux à deux incomparables.

Graphes conceptuels

Les graphes conceptuels sont composés de deux sortes de sommets : les sommets concept et les sommets relation. L'ensemble des arêtes adjacentes à chaque sommet relation r est totalement ordonné, et se représente en numérotant les arêtes de 1 à n , n étant l'arité de la relation r .

Un GC peut être vide, composé d'un unique sommet concept ou encore composé d'un nombre quelconque de sommets concept et sommets relation, mais chaque arête doit obligatoirement relier un sommet relation à un sommet concept, ce qui fait que les GC sont des graphes bipartis.

Définition 2 (Graphe conceptuel) *Un graphe conceptuel, défini sur un support \mathcal{S} , est un 4-uplet $G = (C, R, E, l)$.*

- *(C, R, E) est un multigraphe biparti non-orienté, où C est l'ensemble des sommets concept, R est l'ensemble des sommets relation et E est l'ensemble des arêtes ;*
- *l est une fonction d'étiquetage qui vérifie :*
 - *un sommet concept c est étiqueté par le couple $(type(c), marker(c))$, où $type(c) \subset T_C$ et $marker(c) \subset \mathcal{M}$ (plus spécifiquement, si $marker(c) = *$ le sommet est dit générique, individuel sinon) ;*
 - *un sommet relation r est étiqueté par $type(r) \subset T_R$;*
- *le degré d'un sommet relation r , c'est-à-dire le nombre d'arêtes incidentes à r , est égal à l'arité de $type(r)$;*
- *les arêtes incidentes à un sommet relation r sont totalement ordonnées et sont numérotées de 1 au degré de r ;*
- *le $i^{\text{ème}}$ voisin d'un sommet relation r est donné par $G_i(r)$;*
- *nous noterons $\gamma(r) = (c_1, \dots, c_p)$ le tuple contenant les p voisins d'une relation r d'arité p .*

Un graphe conceptuel se définit toujours par rapport à un support. Nous dirons « soit G un GC défini sur $\mathcal{S} \dots$ » ou, de manière équivalente, « soit $G_{\mathcal{S}}$ un GC \dots ».

1.1.2 Sémantique

Les graphes conceptuels ont une sémantique dans la logique du premier ordre, notée Φ .

Étant donné un support \mathcal{S} , chaque marqueur individuel est associé à une constante; un prédicat unaire est associé à chaque type de concept et un prédicat d'arité n est associé à chaque type de relation d'arité n . Au support lui-même est associé un ensemble de formules $\Phi(\mathcal{S})$ correspondant à l'interprétation des ordres partiels sur T_C et T_R : pour tous types t_1 et t_2 tels que

$t_1 \geq t_2$, la formule associée est $\forall x_1 \dots \forall x_p (t_2(x_1, \dots, x_p) \rightarrow t_1(x_1, \dots, x_p))$, où p est égal à l'arité de la relation pour les types de relation et 1 pour les types de concept.

Tout graphe conceptuel G défini sur \mathcal{S} se voit associer la formule $\Phi(G)$, obtenue de la manière suivante :

- à tout sommet concept on associe un terme :
 - une variable si c'est un sommet concept générique ;
 - la constante associée à son marqueur individuel si c'est un sommet concept individuel
- à tout sommet concept ou relation on associe un atome :
 - à un concept de type t , on associe $t(e)$, où e est le terme associé à t
 - à une relation de type r et voisins c_1, \dots, c_k , on associe $r(e_1, \dots, e_k)$ où e_i est le terme associé à c_i .
- on fait la conjonction de ces atomes et on ferme existentiellement.

Définition 3 *On dit que H est conséquence de G ssi $\Phi(\mathcal{S}), \Phi(G) \models \Phi(H)$.*

1.1.3 Homomorphisme

La notion fondamentale pour tout raisonnement concernant les GC est la relation de généralisation, ou encore subsomption, notée \succeq . Par rapport à la sémantique logique associée au modèle des GC, cette relation correspond à l'implication logique. Nous allons définir la relation \succeq par un homomorphisme de graphes.

Définition 4 (Relation d'ordre sur les sommets) *La relation d'ordre sur les sommets concept se traduit de la manière suivante :*

$$(c, m) \leq (c', m') \text{ ssi } c \leq c' \text{ et } m \leq m'$$

où $c, c' \in T_C$ et $m, m' \in \mathcal{M}$.

La relation d'ordre sur les sommets relation est, tout simplement, celle de T_R . Les sommets relation ne sont comparables que s'ils ont la même arité.

Définition 5 (Homomorphisme) *Un homomorphisme d'un GC $H = (R_H, C_H, E_H, l_H)$ dans un GC $G = (R_G, C_G, E_G, l_G)$, définis sur le même support \mathcal{S} , est une application Π de C_H dans C_G qui vérifie les propriétés suivantes :*

1. *peut restreindre les étiquettes des sommets : pour tout sommet $c \in C_H$, $l_H(c) \geq l_G(\Pi(c))$;*
2. *pour tout sommet $r \in R_H$, avec $\gamma(r) = (x_1, \dots, x_p)$, où x_i est le $i^{\text{ème}}$ voisin de r , il existe $r' \in R_G$ avec $\gamma(r') = (\Pi(x_1), \dots, \Pi(x_p))$ et $l_H(r) \geq l_G(r')$.*

Soient H et G deux GC définis sur \mathcal{S} . Quand un homomorphisme de H vers G existe, nous pouvons dire que H subsume G ou que H généralise G , que l'on note $H \succeq_{\mathcal{S}} G$. Réciproquement, nous pouvons aussi dire que G est subsumé par H ou que G spécialise H , ce que l'on note $G \preceq_{\mathcal{S}} H$.

Un homomorphisme se définit toujours par rapport à un support donné. Nous pouvons dire « soit Π un homomorphisme de H vers G , deux GC définis sur $\mathcal{S} \dots$ », ou alors « soit Π un \mathcal{S} -homomorphisme de H vers $G \dots$ ». Il nous arrivera de parler de *projection* à la place d'homomorphisme.

Théorème 1 *L'homomorphisme est adéquat et complet par rapport à la logique du premier ordre. Étant donnés deux GC définis sur \mathcal{S} , si $H \succeq_{\mathcal{S}} G$ alors $\Phi(\mathcal{S}), \Phi(G) \models \Phi(H)$ (adéquation [Sow84]). La complétude [CM92], c'est-à-dire $\Phi(\mathcal{S}), \Phi(G) \models \Phi(H)$ implique $H \succeq_{\mathcal{S}} G$, est vérifiée si G est en forme normale.*

Définition 6 (Forme normale) *Un GC sous forme normale est un GC dans lequel chaque marqueur individuel apparaît au plus une fois. Si deux sommets concept c et c' portent le même marqueur individuel, ils sont alors fusionnés en un seul sommet concept qui aura comme type $\text{type}(c)$ si $c \leq c'$, $\text{type}(c')$ sinon.*

1.2 Bases de données

Informellement, une base de données (BD) est un ensemble de *tables*, chaque table regroupant les éléments ayant une signification commune. Les *requêtes* sont utilisées pour extraire des informations des tables.

1.2.1 Syntaxe

Terminologie

Sauf indication contraire, les ensembles considérés sont infinis dénombrables. Nous supposons donnés :

- un ensemble d'*attributs* **att** ;
- un ensemble de constantes, disjoint du précédent et appelé *domaine* et noté **dom** ;
- un ensemble de noms de relations **relname**, disjoint des précédents ;

- chaque nom de relation est associé à un tuple *fini* d'attributs distincts. Cette association, dite *sorte*¹ de R pour une relation appelée R , est notée $sort : \mathbf{relname} \mapsto \mathcal{P}^{\text{finie}}(\mathbf{att})$;
- l'arité d'une relation R se note *arity* et nous avons $arity(R) = |sort(R)|$.

Tables et bases de données

Définition 7 (Schéma relationnel) *Un schéma relationnel $R[U]$ est donné par une relation $R \in \mathbf{relname}$ et sa sorte $U = sort(R)$.*

Définition 8 (Schéma de base de données) *Un schéma de bases de données est un ensemble fini non vide de schémas relationnels $\mathbf{R} = \{R_1[U_1], R_2[U_2], \dots, R_k[U_k]\}$.*

Une instance I d'un schéma relationnel $R[U]$ est un ensemble *fini* de tuples définis sur $R[U]$. De même, une instance \mathbf{I} d'un schéma de BD \mathbf{R} est une application ayant pour domaine \mathbf{R} et telle que $\forall R_i \in \mathbf{R} : \mathbf{I}(R_i)$ est une instance de R .

Une *relation*² est la donnée d'un schéma relationnel $R[U]$ et d'un ensemble fini de tuples définis sur \mathbf{dom}^k , où $k = arity(R)$. De même, une *base de données*³ est la donnée d'un schéma de base de données \mathbf{R} et d'un ensemble fini non vide de relations.

Requêtes conjonctives positives

Une requête est une application depuis l'ensemble des instances d'un schéma (le schéma d'entrée) vers l'ensemble des instances d'un schéma (le schéma de sortie), où *schéma* peut être un schéma de base de données ou de relations.

Deux requêtes q_1, q_2 seront dites équivalentes si elles sont définies depuis le même schéma \mathbf{R} vers le même schéma \mathbf{S} et pour toute instance I de \mathbf{R} : $q_1(I) = q_2(I)$.

Nous allons maintenant introduire un ensemble **var** dénombrable de *variables*, disjoint de tous les autres vus jusqu'à présent, qui sera interprété (valué) sur **dom**.

Un *terme* est un élément de $\mathbf{dom} \cup \mathbf{var}$.

Avec l'ensemble **var**, nous généralisons la notion de tuple de $R[U]$ par une application de U dans $\mathbf{dom} \cup \mathbf{var}$, ainsi que la notion d'atomes de R par une

¹Cependant, rien n'empêche que deux relations différentes aient la même sorte

²Ou, plus exactement, une *instance d'un schéma relationnel*

³Ou encore *instance d'un schéma de base de données*

expression de la forme $R(t_1, \dots, t_n)$ où n est l'arité de R et les t_i sont des termes. Un atome de base (ou fait) est un atome dont tous les termes sont des constantes.

On interprète un ensemble de variables $V = (v_1, \dots, v_n)$ en définissant une application $\mathbf{val} : V \mapsto \mathbf{dom}$. Cette définition peut être naturellement étendue aux termes en valuant les constantes par elles-mêmes.

Définition 9 (Requête conjonctive) Soit \mathbf{R} un schéma de BD. Une requête sous forme de règle conjonctive — ou simplement règle — sur \mathbf{R} est une expression de la forme :

$$\mathit{ans}(u) \leftarrow R_1(u_1), \dots, R_n(u_n)$$

telle que :

- $n \geq 0$, R_1, \dots, R_n sont des noms de relation de \mathbf{R}
- ans n'est pas un nom de relation de \mathbf{R} ,
- u, u_1, \dots, u_n sont des tuples libres (peuvent contenir des variables et des constantes) d'arité cohérente avec celles de R_1, \dots, R_n ; u peut être d'arité 0
- chaque variable de u doit avoir une occurrence dans u_1, \dots, u_n

La partie à gauche de " \leftarrow " est la tête de la requête, et la partie à droite est le corps.

Aussi, nous notons $\mathit{var}(q)$ l'ensemble des variables présentes dans une requête q .

1.2.2 Sémantique

Étant donné un sous-ensemble fini V de \mathbf{var} , une *interprétation* (ou *valuation*) v sur V est une fonction de V sur \mathbf{dom} .

Intuitivement, les règles sont des outils de *déduction* : s'il est possible de trouver des « valeurs » pour les variables telles que, avec ces valeurs, le corps de la règle soit constitué de faits de l'instance du schéma, alors il est possible de déduire la tête comme un fait.

Plus formellement :

Définition 10 La sémantique d'une requête sous forme de règles q est définie par :

$$q(\mathbf{I}) = \{v(u) \mid v \text{ est une valuation sur } \mathit{var}(q) \text{ et } \forall i \in [1, n] : v(u_i) \in \mathbf{I}(R_i)\}$$

On appelle parfois les relations du corps comme faisant partie de la BD *extensionnelle*, alors que la relation de la tête est une relation *intentionnelle*, non stockée mais calculée.

1.2.3 Réponse à une requête

Une valuation v est une fonction $v : var(Q) \mapsto constantes$, qu'on étend naturellement aux constantes par $v : constantes \mapsto constantes$ (dans ce cas, v est l'identité).

Une valuation v d'une requête Q sur une base de données B satisfait B si $\forall R(x_1, \dots, x_n)$ dans le corps de Q , $v(x_1), \dots, v(x_n) \in R$.

L'ensemble des solutions à Q dans B est la projection sur les variables de la tête de Q de l'ensemble des valuations Q qui satisfont B .

1.2.4 Notations

Pour tout ce qui concerne les bases de données, les notations suivantes seront utilisées :

Constantes	a, b, c
Variables	x, y
Ensembles de variables	X, Y
Termes	e
Attributs	A, B, C
Ensembles d'attributs	U, V, W
Noms de relations	$R, S; R[U], S[V]$
Bases de données	R, S
Tuples	t, s
Tuples libres	u, v, w
Faits	$R(a_1, \dots, a_n), R(t)$
Atomes	$R(e_1, \dots, e_n), R(u)$
Instances de relations	I, J
Instances de bases de données	I, J

Chapitre 2

Transformations et équivalences

2.1 Bases de données avec support

Dans ce chapitre nous montrons comment passer d'un problème de projection entre graphes conceptuels à un problème de requête sur une base de données, et inversement.

Nous introduisons aussi un modèle inédit, que nous appellerons *base de données avec support* (BD + \mathcal{S}).

Nous détaillerons toutes les transformations nécessaires pour transformer ces modèles en prouvant leur validité.

2.1.1 Syntaxe

Le modèle présenté ici est une extension du modèle base de données, auquel on ajoute un support, défini de la même manière que dans la partie graphes conceptuels.

Dans ce nouveau modèle, que nous appellerons simplement *base de données avec support* (BD + \mathcal{S}), le mécanisme de réponse ne peut pas être le même que dans le modèle BD, car il doit pouvoir prendre en compte les informations supplémentaires disponibles dans le support, comme l'ordre sur les concepts et types. Les définitions des tables et des requêtes, ainsi que les notations, sont les mêmes que celles données dans la partie BD.

2.1.2 Réponse à une requête

La réponse à une requête dans le modèle BD + \mathcal{S} , que nous appellerons \mathcal{S} -réponse pour une base de données définie sur un support \mathcal{S} , doit être en mesure d'exploiter les informations contenues sur le support car, à la

différence du modèle BD présenté ci-dessus, la base de données, dans ce modèle-ci, ne contient pas toutes les informations.

En analysant simultanément la base de données et le support associé, il est possible d'obtenir des informations « nouvelles », qui ne sont pas directement exprimées dans la base de données.

La *valuation* se définit de la même manière que dans le modèle BD. Mais ici, une valuation v d'une requête Q sur une base de données B définie sur un support \mathcal{S} satisfait B si $\forall R(x_1, \dots, x_n)$ dans le corps de Q , $\exists R'$ tel que $R \geq R'$ et $v(x_1), \dots, v(x_n) \in R'$, où \geq est la relation d'ordre définie sur le support.

L'ensemble des solutions à Q dans B est la projection sur les variables de la tête de Q de l'ensemble des valuations de Q qui satisfont B .

2.2 Équivalence entre GC et BD + \mathcal{S}

Nous montrons ici l'équivalence entre les graphes conceptuels et les bases de données avec support.

2.2.1 Transformations

Définition 11 (Graphe vers requête) Soit $H = (C, R, E, l)$ un graphe conceptuel défini sur un support \mathcal{S} . La requête conjonctive $q(H)$ se construit de la manière suivante :

1. notons μ la fonction qui, pour tout sommet concept c retourne $\text{marker}(c)$ si ce n'est pas un marqueur générique, et une variable unique (et toujours la même pour ce même sommet concept) dans le cas contraire ;
2. $\forall c \in C$, avec $t = \text{type}(c)$ et $m = \mu(c)$, il existe un unique atome dans le corps de $q(H)$ de la forme $t(m)$;
3. $\forall r \in R$ avec $t = \text{type}(r)$, $\gamma(r) = (c_1, \dots, c_p)$ et $m_1, \dots, m_p = \mu(c_1), \dots, \mu(c_p)$, il existe un unique atome dans le corps de $q(H)$ de la forme $t(m_1, \dots, m_p)$.
4. la tête de q est définie par $\text{ans}(x_1, \dots, x_p)$, où x_1, \dots, x_p sont les variables retournées par μ au cours de la création du corps de q .

Propriété 1 Si H est un GC et $q(H)$ est la requête conjonctive associée à H , alors les sommets concept de H sont en bijection avec les « atomes concept » de $q(H)$. De la même manière, les sommets relation de H sont en bijection avec les « atomes relation » de $q(H)$.

Définition 12 (Graphe vers BD + \mathcal{S}) Soit $G = (C, R, E, l)$ un graphe conceptuel défini sur un support \mathcal{S} . La base de données $b(G)$ se construit de la manière suivante :

1. notons μ' la fonction qui, pour tout sommet concept c retourne $\text{marker}(c)$ si ce n'est pas un marqueur générique, et une constante unique (et toujours la même pour ce même sommet concept) dans le cas contraire ;
2. notons $T_C(G)$ l'ensemble des types de concept intervenant dans G :
 $T_C(G) = \{t \in T_C \mid \exists c \in C_G, \text{type}(c) = t\}$;
3. $\forall t \in T_C(G)$, créer une table t dans $b(G)$ d'arité 1 ;
4. $\forall c \in C$ avec $t = \text{type}(c)$, insérer le tuple $(\mu'(c))$ dans la table t ;
5. notons $T_R(G)$ l'ensemble des types de relation intervenant dans G :
 $T_R(G) = \{t \in T_R \mid \exists r \in R_G, \text{type}(r) = t\}$;
6. $\forall t \in T_R(G)$, créer une table t dans $b(G)$ de même arité que r ;
7. $\forall r \in R_G$ avec $\gamma(r) = (c_1, \dots, c_p)$ et $t = \text{type}(r)$, insérer le tuple $(\mu'(c_1), \dots, \mu'(c_p))$ dans la table t .

Propriété 2 Si G est un GC et $b(G)$ est la base de données associée à G , alors les sommets concept de G sont en bijection avec les « tuples concept » de $b(G)$. De la même manière, les sommets relation de G sont en bijection avec les « tuples relation » de $b(G)$.

2.2.2 Théorème

Théorème 2 Soient G et H deux graphes conceptuels définis sur un support \mathcal{S} .

$$H \succeq_{\mathcal{S}} G \Leftrightarrow \text{il existe une } \mathcal{S}\text{-réponse à } q(H) \text{ dans } b(G)$$

Preuve

(\Rightarrow) Soit Π une \mathcal{S} -projection de H dans G , soit $\mathcal{A} = \mu' \circ \Pi \circ \mu^{-1}$ une application de $q(H)$ dans $b(G)$.

Montrons que \mathcal{A} est une \mathcal{S} -réponse de $q(H)$ dans $b(G)$. Commençons par les atomes « concept », c'est-à-dire ceux de la forme $a(x)$ où a est un type de concept.

1. soit t un atome de $q(H)$ de la forme $a(x)$, où $a \in T_C$;
2. $\exists! c \in C_H$ tel que $\text{type}(c) = a$ et $\mu(c) = x$;
3. Π est une projection, donc $\exists \Pi(c) \in C_G$ tel que $\text{type}(c) \geq \text{type}(\Pi(c))$ et $\text{marker}(c) \geq \text{marker}(\Pi(c))$;

4. si $\Pi(c) \in C_G$, alors il existe un unique atome dans $b(H)$ de la forme $a'(x')$ tel que $type(\Pi(c)) = a'$ et $\mu'(\Pi(c)) = x'$ (si $marker(c)$ est une constante, alors $x' = marker(c)$, sinon x' est une constante générée par μ');
5. l'image de $a(x)$ par $\mu' \circ \Pi \circ \mu^{-1}$, c'est-à-dire par \mathcal{A} , est $a'(x')$ tel que $a \geq a'$ et $x = x'$ si x est une constante, sinon x' est la constante associée à la variable x .

Et maintenant, les « atomes relation » :

1. soit t un atome de $q(H)$ de la forme $a(x_1, \dots, x_p)$, où $a \notin T_C$;
2. $\exists! r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$ tel que $type(r) = a$ et $\mu(c_1), \dots, \mu(c_p) = x_1, \dots, x_p$;
3. Π étant une projection, $\exists r' \in R_G$ avec $\gamma(r') = \Pi(c_1), \dots, \Pi(c_p)$ et $type(r) \geq type(r')$;
4. si $r' \in R_G$, alors il existe un unique atome dans $b(G)$ de la forme $a'(x'_1, \dots, x'_p)$ tel que $type(r') = a'$ et $\mu'(\Pi(c_1)), \dots, \mu'(\Pi(c_p)) = x'_1, \dots, x'_p$;
5. nous avons donc $a \geq a'$ et, $\forall i \in [1; p]$, x_i a pour image x'_i qui est une constante.

Nous avons donc montré que \mathcal{A} est une valuation qui valide $b(G)$. \mathcal{A} est donc une \mathcal{S} -réponse de $q(H)$ dans $b(G)$. Vérifions l'autre sens de l'équivalence.

(\Leftarrow) Soit \mathcal{A} une \mathcal{S} -réponse de $q(H)$ dans $b(G)$ et soit Π l'application $\Pi = \mu'^{-1} \circ \mathcal{A} \circ \mu$. Montrons que Π est une \mathcal{S} -projection de H dans G .

- Π est bien une application de C_H dans C_G
- Montrons que $\forall c \in C_H, l_H(c) \geq l_G(\Pi(c))$:

1. soit $c \in C_H$, $\exists! t \in q(H)$ de la forme $a(x)$, où $a = type(c)$ et $\mu(c) = x$;
2. \mathcal{A} étant une réponse, $\exists t' \in b(H)$ de la forme $a'(x')$, où $a \geq a'$ et x' est une constante ($x = x'$ si x est une constante);
3. $t' \in b(G)$, donc $\exists! c' \in C_G$ tel que $\mu'(c') = x'$ et $type(c') = a'$;
4. donc $type(c) \geq type(c')$ et $marker(c) \geq marker(c')$, d'où $l_H(c) \geq l_G(c')$.

- Montrons maintenant que $\forall r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$, il existe $r' \in R_G$ avec $\gamma(r') = (\Pi(c_1), \dots, \Pi(c_p))$ et $type(r) \geq type(r')$.

1. soit $r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$, alors $\exists! t \in q(H)$ de la forme $a(x_1, \dots, x_p)$ avec $type(r) = a$ et $x_1, \dots, x_p = \mu(c_1), \dots, \mu(c_p)$;

2. \mathcal{A} est une \mathcal{S} -réponse, donc $\exists t' \in b(G)$ de la forme $a'(x'_1, \dots, x'_p)$ avec $a \geq a'$ et $x'_1, \dots, x'_p = \mathcal{A}(x_1), \dots, \mathcal{A}(x_p)$;
3. $t' \in b(G)$, donc $\exists! r' \in R_G$ avec $\gamma(r') = c'_1, \dots, c'_p$, où $\text{type}(r') = a'$ et $c_1, \dots, c_p = \mu'^{-1}(x_1), \dots, \mu'(x_p)$;
4. donc $\text{type}(r) \geq \text{type}(r')$ et $\gamma(r) = \Pi(c_1), \dots, \Pi(c_p)$.

Π est donc une \mathcal{S} -projection de H dans G . □

2.3 Élimination du support d'un GC

Pour pouvoir éliminer le support d'un graphe conceptuel, nous avons besoin d'effectuer certaines opérations sur le support et le graphe lui-même. Dans un premier temps, nous éliminerons les types de concept; ensuite, nous nous débarrasserons de la hiérarchie entre les types de concept et de relation.

2.3.1 Réification

Cette transformation a pour but d'éliminer les types de concept, pour ne garder que \top , obtenant ainsi un support sans type de concept, *purement relationnel*.

Définition 13 (Support purement relationnel) *Un support \mathcal{S} où $T_C = \{\top\}$ est un support purement relationnel*

δ : relationnalisation d'un support Soit $\mathcal{S} = (T_C, \{T_R^1, T_R^2, \dots, T_R^k\}, \mathcal{M})$ un support. Le support purement relationnel $\delta(\mathcal{S}) = (\{\top\}, \{T_R^1, T_R^2, \dots, T_R^k\}, \mathcal{M})$ associé à \mathcal{S} est défini de la façon suivante (nous pouvons voir un exemple à la figure 2.1) :

- $T_R^{1'} = T_C \cup T_R^1 \cup \{\top_N\}$, où \top_N est le nouvel élément maximal;
- $\forall t \in T_R^{1'}, t \leq_N \top_N$;
- $\forall x, y \in T_C, x \leq_C y \Leftrightarrow x \leq_N y$;
- $\forall x, y \in T_R^1, x \leq_1 y \Leftrightarrow x \leq_N y$;
- si $x \in T_C$ et $y \in T_R^1$, alors x et y sont incomparables.

δ : relationnalisation d'un graphe Soit $G = (C, R, E, l)$ un graphe conceptuel défini sur un support \mathcal{S} . Le graphe conceptuel relationnalisé $\delta(G) = (C', R', E', l')$, défini sur un support purement relationnel, s'obtient de la manière suivante (voir fig. 2.2 pour un exemple) :

- $\forall c \in C, \exists! c' \in C'$ tel que $l'(c') = (\top, \text{marqueur}(c))$; le sommet concept c' ainsi défini est l'image de c par δ , que nous noterons $\delta(c)$;

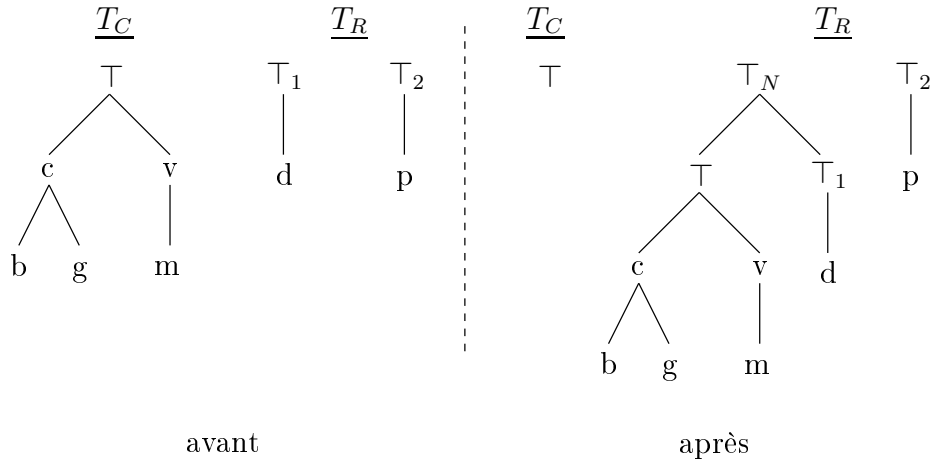


FIG. 2.1 – Réification d'un support

- $\forall c \in C, \exists !r' \in R'$ tel que $l'(r') = type(c)$ et $\gamma(r') = (\delta(c))$;
- $\forall r \in R$ avec $\gamma(r) = (c_1, \dots, c_p)$, $\exists !r' \in R'$ tel que $l'(r') = l(r)$ et $\gamma(r') = (\delta(c_p), \dots, \delta(c_p))$.

Propriété 3 *Par construction, les sommets concept d'un graphe G sont en bijection avec les sommets concept de $\delta(G)$.*

Lemme 1 *Si G est un GC défini sur \mathcal{S} alors $\delta(G)$ est un GC défini sur $\delta(\mathcal{S})$.*

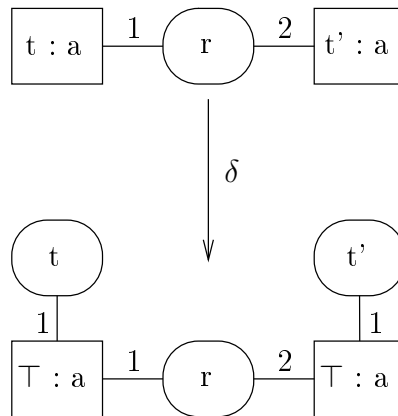


FIG. 2.2 – Réification d'un graphe

Ceci nous mène au théorème suivant :

Théorème 3 Soient G et H deux graphes conceptuels définis sur un support \mathcal{S} . Alors :

$$H \succeq_{\mathcal{S}} G \Leftrightarrow \delta(H) \succeq_{\delta(\mathcal{S})} \delta(G)$$

Preuve

(\Rightarrow) Soit Π une projection de H dans G . Montrons que $\Pi' = \delta \circ \Pi \circ \delta^{-1}$ est une $\delta(\mathcal{S})$ -projection de $\delta(H)$ dans $\delta(G)$ (la figure 2.3 p. 19 représente le schéma de cette preuve) :

La notation $\delta(H) \succeq_{\delta(\mathcal{S})} \delta(G)$ a bien un sens (lemme 1). Si $c \in C_{\delta(H)}$, alors $\delta^{-1}(c)$ est un sommet unique de H (δ est bijective, propriété 3). $\Pi \circ \delta^{-1}(c) \in C_G$ et $\delta \circ \Pi \circ \delta^{-1}(c) = \Pi'(c) \in C_{\delta(G)}$. Π' est donc bien une application de $C_{\delta(H)}$ dans $C_{\delta(G)}$.

Montrons que $\forall c \in C_{\delta(H)}, l_{\delta(H)}(c) \geq l_{\delta(G)}(\Pi'(c))$, c'est-à-dire que $type(c) \geq type(\Pi'(c))$ et que $marker(c) \geq marker(\Pi'(c))$:

1. si $c \in C_{\delta(H)}$, alors $\exists! \delta^{-1}(c) \in C_H$ tel que $marker(c) = marker(\delta^{-1}(c))$;
2. Π étant une projection, $\exists \Pi \circ \delta^{-1}(c) \in C_G$ tel que $marker(\delta^{-1}(c)) \geq marker(\Pi \circ \delta^{-1}(c))$
3. comme $\Pi \circ \delta^{-1}(c) \in C_G, \exists! \delta \circ \Pi \circ \delta^{-1}(c) \in C_{\delta(G)}$ tel que $marker(\delta \circ \Pi \circ \delta^{-1}(c)) = marker(\Pi \circ \delta^{-1}(c))$;
4. donc $marker(c) \geq marker(\Pi'(c))$;
5. en outre, $type(c) = type(\Pi'(c))$ (le seul type étant \top , par construction de $\delta(G)$ et $\delta(H)$), alors $l_{\delta(H)}(c) \geq l_{\delta(G)}(\Pi'(c))$.

Montrons maintenant que $\forall r \in R_{\delta(H)}$ avec $\gamma(r) = (c_1, \dots, c_p)$, il existe $r' \in R_{\delta(G)}$ avec $\gamma(r') = (\Pi'(c_1), \dots, \Pi'(c_p))$ et $l_{\delta(H)}(r) \geq l_{\delta(G)}(r')$. δ ayant créé des sommets relation « inédits », qui ne sont pas présents dans H , nous allons démontrer cette propriété en deux étapes. Commençons par les sommets relation dits « inédits » :

1. soit $r \in R_{\delta(H)}$, vérifiant $type(r) \in T_R^1$ et $type(r) \leq \top$ (c'est une relation construite à partir d'un sommet concept, donc nécessairement unaire), avec $\gamma(r) = c$, où $type(c) = \top$;
2. alors $\exists! \delta^{-1}(c) \in C_H$ tel que $type(\delta^{-1}(c)) = type(r)$ et $marker(\delta^{-1}(c)) = marker(c)$;
3. Π étant une projection, $\exists \Pi \circ \delta^{-1}(c) \in C_G$ tel que $type(\delta^{-1}(c)) \geq type(\Pi \circ \delta^{-1}(c))$ et $marker(\delta^{-1}(c)) \geq marker(\Pi \circ \delta^{-1}(c))$;
4. $\Pi \circ \delta^{-1}(c)$ appartenant à $C_G, \exists! \delta \circ \Pi \circ \delta^{-1}(c)$ avec $type(\delta \circ \Pi \circ \delta^{-1}(c)) = \top$ et $marker(\Pi \circ \delta^{-1}(c)) = marker(\delta^{-1}(c))$ et, par construction, $\exists! r' \in R_{\delta(G)}$ avec $\gamma(r') = \delta \circ \Pi \circ \delta^{-1}(c)$ et $type(r') = type(\Pi \circ \delta^{-1}(c))$;

5. nous avons ainsi $type(r) \geq type(r')$ et $\gamma(r') = \Pi'(c)$; les sommets relation créés par δ vérifient donc la propriété de voisinage.

Attaquons-nous maintenant aux autres sommets relation :

1. soit $r \in R_{\delta(H)}$, vérifiant $type(r) \not\leq \top$ (c'est une relation qui existait déjà dans H), avec $\gamma(r) = c_1, \dots, c_p$;
2. par construction, $\exists! \delta^{-1}(r) \in R_H$ tel que $\gamma(\delta^{-1}(c)) = \delta^{-1}(c_1), \dots, \delta^{-1}(c_p)$;
3. Π étant une projection, il existe $r' \in R_G$ tel que $\gamma(r') = \Pi \circ \delta^{-1}(c_1), \dots, \Pi \circ \delta^{-1}(c_p)$ et $l_H(r) \geq l_G(r')$
4. r' appartenant à R_G , $\exists! r'' = \delta(r') \in R_{\delta(G)}$ avec $\gamma(\delta(r')) = \delta \circ \Pi \circ \delta^{-1}(c_1), \dots, \delta \circ \Pi \circ \delta^{-1}(c_p)$ et $l_G(r') = l_{\delta(G)}(\delta(r'))$
5. il vient que $l_H(r) \geq l_{\delta(G)}(r'')$ et $\gamma(r'') = \Pi'(c_1), \dots, \Pi'(c_p)$; les sommets relation qui n'ont pas été créés par δ vérifient eux aussi la propriété de voisinage.

Nous prouvons ainsi que Π' est une $\delta(\mathcal{S})$ -projection de $\delta(H)$ dans $\delta(G)$.

(\Leftarrow) Nous devons maintenant montrer que le théorème se vérifie aussi dans l'autre sens.

Soit Π' une $\delta(\mathcal{S})$ -projection de $\delta(H)$ dans $\delta(G)$, montrons que l'application $\Pi = \delta^{-1} \circ \Pi' \circ \delta$ est une \mathcal{S} -projection de H dans G , en commençant par la propriété sur les sommets concept :

1. si $c \in C_H$, $\exists! \delta(c) \in C_{\delta(H)}$ adjacent à un sommet relation r tels que $marker(x) = marker(\delta(c))$, $type(c) = type(r)$ et $\gamma(r) = \delta(c)$;
2. Π' étant une projection, $\exists \Pi' \circ \delta(c) \in C_{\delta(G)}$ tel que $marker(\delta(c)) \geq marker(\Pi' \circ \delta(c))$ et $\exists r' \in R_{\delta(G)}$ avec $\gamma(r') = \Pi' \circ \delta(c)$ et $type(r') = type(r)$;
3. par construction, $\exists! \delta^{-1} \circ \Pi' \circ \delta(c) \in C_G$ tel que $marker(\delta^{-1} \circ \Pi' \circ \delta(c)) = marker(\Pi' \circ \delta(c))$ et $type(\delta^{-1} \circ \Pi' \circ \delta(c)) = type(r')$;
4. nous avons donc un sommet relation appartenant à C_G tel que $type(c) \geq type(\Pi(c))$ et $marker(c) \geq marker(\Pi(c))$, d'où $l_H(c) \geq l_G(\Pi(c))$.

Nous devons à présent vérifier la propriété concernant les sommets relation :

1. si $r \in R_H$, avec $\gamma(r) = (c_1, \dots, c_p)$, $\exists! \delta(r) \in R_{\delta(H)}$ tel que $type(r) = type(\delta(r))$ et $\gamma(\delta(r)) = \delta(c_1), \dots, \delta(c_p)$;
2. comme Π' est une projection, $\exists r' \in R_{\delta(G)}$ tel que $\gamma(r') = \Pi' \circ \delta(c_1), \dots, \Pi' \circ \delta(c_p)$ et $marker(\delta(r)) \geq marker(r')$;

3. par construction, $\exists! r'' = \delta^{-1}(r') \in R_G$ tel que $\gamma(r'') = \delta^{-1} \circ \Pi' \circ \delta(c_1), \dots, \delta^{-1} \circ \Pi' \circ \delta(c_p)$ et $\text{type}(r'') = \text{type}(r')$;
4. ainsi, $\text{type}(r) \geq \text{type}(r'')$ et $\gamma(r'') = \Pi(c_1), \dots, \Pi(c_p)$.

Ceci conclut cette démonstration ; Π est donc bien une \mathcal{S} -projection de H dans G . \square

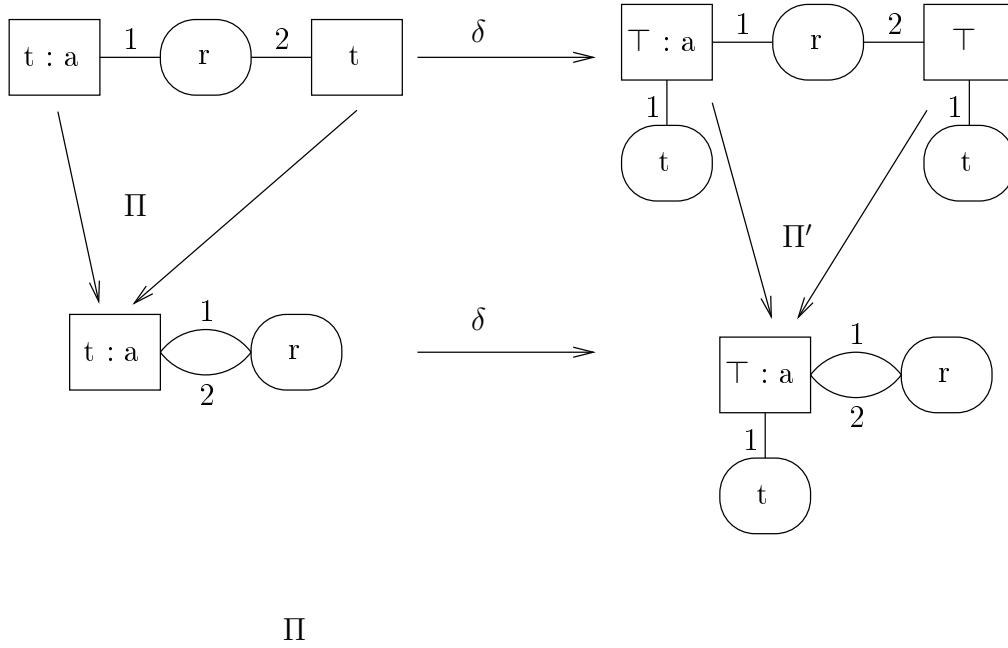


FIG. 2.3 – Schéma de la preuve de la réification

2.3.2 Aplatissement et expansion

La deuxième étape dans l'élimination du support est la suppression de la hiérarchie des types dans le support. Comme dans la section précédente, nous utilisons deux transformations : l'*aplatissement*, qui supprime la hiérarchie dans le support et l'*expansion*, qui modifie le graphe pour qu'il représente *explicitement* toute l'information qui était jusqu'alors *implicite*, puisque retrouvée avec l'aide du support.

Définition 14 (Ordre partiel plat) *Un ordre partiel est dit plat si tous ses éléments sont deux à deux incomparables.*

Définition 15 (Support plat) *Un support $\mathcal{S} = (T_C, \{T_R^1, \dots, T_R^k\}, \mathcal{M})$ est dit plat si T_C, T_R^1, \dots, T_R^k sont plats.*

Un support plat ne contient aucune information nécessaire au raisonnement :

Proposition 1 *Si \mathcal{S} est plat, $H \succeq_{\mathcal{S}} G$ ssi $\Phi(G) \models \Phi(H)$.*

Preuve C'est un cas particulier du théorème 1. Il suffit de voir que $\Phi(\mathcal{S})$ est la formule vide lorsque \mathcal{S} est plat. \square

Nous montrons ici qu'il est possible d'intégrrer à un graphe toutes les informations nécessaires au raisonnement contenues dans un support.

Bien que l'aplatissement d'un support et l'expansion d'un graphe soient deux opérations différentes, elles vont de pair, et c'est pourquoi nous utiliserons la même notation pour les deux, sans risque d'ambiguïté.

Nous avons vu ce qu'est un support plat (déf. 15). Nous allons voir ici comment, à partir d'un support *sans type de concept* (déf. 13), obtenir un support plat.

α : aplatissement d'un support Soit $\mathcal{S} = (\{\top\}, \{T_R^1, \dots, T_R^k\}, \mathcal{M})$ un support sans type de concept, avec les relations d'ordre \leq_1 sur T_R^1, \dots, \leq_k sur T_R^k . Le support aplati construit à partir de \mathcal{S} , que nous noterons $\alpha(\mathcal{S})$, s'obtient en éliminant les ordres partiels sur T_R^1, \dots, T_R^k , de sorte que T_R^1, \dots, T_R^k soient des ordres partiels plats.

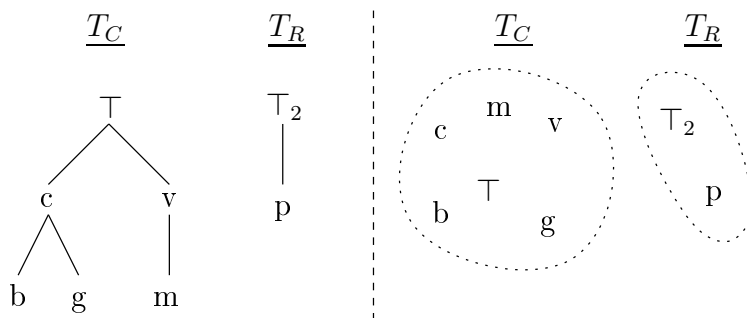


FIG. 2.4 - $\alpha(\mathcal{S})$ - à gauche, le support original, à droite le support aplati

Maintenant que nous savons comment aplatir un support, regardons comment expander un graphe. De manière informelle, le but de cette transformation est de représenter dans le graphe toute l'information qui peut être déduite du graphe et de son support, de manière à ce que tous les raisonnements qui étaient possibles avec le support soient aussi possibles sans celui-ci.

α : expansion d'un graphe Soit $G = (C, R, E, l)$ un graphe conceptuel défini sur un support \mathcal{S} sans type de concept. Le graphe conceptuel $\alpha(G) = (C, R', E', l')$, défini sur le support $\alpha(\mathcal{S})$, s'obtient de la manière suivante : $\forall r \in R$, avec $\gamma(r) = (c_1, \dots, c_k)$ et k étant l'arité de r :

- soit P l'ensemble des types tels que $\forall p \in P, \text{type}(r) \leq p$, sans considérer les éléments \top et \top_N dans le cas d'une relation unaire ;
- alors pour tout p dans P , il existe un unique sommet relation r' dans R' avec $\gamma(r') = (c_1, \dots, c_k)$.

Propriété 4 Si G est un graphe conceptuel défini sur \mathcal{S} , alors G est aussi défini sur $\alpha(\mathcal{S})$.

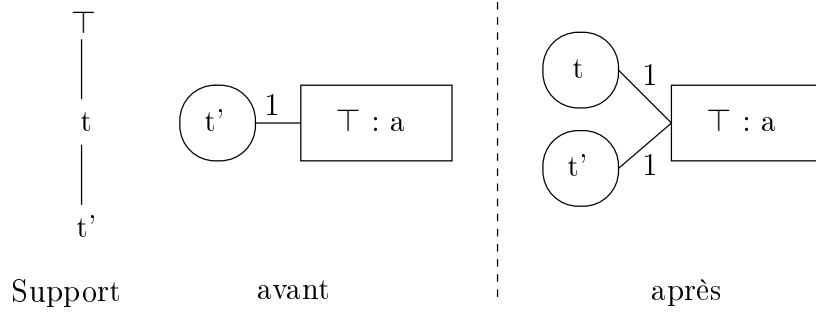


FIG. 2.5 – Graphe expansé

Proposition 2 Un graphe conceptuel H et sa version expansée vérifient la propriété suivante :

$$H \equiv_{\mathcal{S}} \alpha(H)$$

ou, en d'autres mots :

$$H \succeq_{\mathcal{S}} \alpha(H) \text{ et } \alpha(H) \succeq_{\mathcal{S}} H$$

Preuve Immédiat :

(\Rightarrow) Par construction, $H \subseteq \alpha(H)$;

(\Leftarrow) α ne rajoute que des sommets qui soient *plus génériques* que ceux présents dans H ; ces sommets-là peuvent donc être projetés sur leur « sommet d'origine », c'est-à-dire celui depuis lequel ils furent créés. \square

Propriété 5 *L'expansion d'un graphe est une opération idempotente :*

$$\alpha(H) \Leftrightarrow \alpha \circ \alpha(H)$$

Théorème 4 *Soient G et H deux graphes conceptuels définis sur un support \mathcal{S} purement relationnel, alors :*

$$H \succeq_{\mathcal{S}} G \Leftrightarrow H \succeq_{\alpha(\mathcal{S})} \alpha(G)$$

Preuve

(\Rightarrow) Soit Π une \mathcal{S} -projection de H dans G . Montrons que $\Pi' = \alpha \circ \Pi$ est une $\alpha(\mathcal{S})$ -projection de H dans $\alpha(G)$:

- Π' est bien une application de C_H dans $C_{\alpha(G)}$
- Montrons que $\forall c \in C_H, l_H(c) \geq l_{\alpha(G)}(\Pi'(c))$:
 1. soit $c \in C_{\alpha(H)}$;
 2. Π étant une projection, $\exists \Pi(c) \in C_G$ tel que $type(c) = type(\Pi(c))$ (car T_C ne contient que \top) et $marker(c) \geq marker(\Pi(c))$;
 3. $\Pi(c) \in C_G$, donc $\exists ! \alpha \circ \Pi(c) \in C_{\alpha(G)}$ tel que $type(\Pi(c)) = type(\alpha \circ \Pi(c))$ et $marker(\Pi(c)) = marker(\alpha \circ \Pi(c))$;
 4. comme $\alpha \circ \Pi(c) = \Pi'(c)$, il vient que $l_H(c) \geq l_{\alpha(G)}(\Pi'(c))$.
- Montrons maintenant que $\forall r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$, il existe $r' \in R_{\alpha(G)}$ avec $\gamma(r') = (\Pi'(c_1), \dots, \Pi'(c_p))$ et $l_H(r) \geq l_{\alpha(G)}(r')$.
 1. soit $r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$;
 2. Π est une projection, donc $\exists s \in R_G$ tel que $type(r) \geq type(s)$ et $\gamma(s) = \Pi(c_1), \dots, \Pi(c_p)$;
 3. $s \in R_G$ alors, par construction, $\forall t \in T_R^p$ tel que $type(s) \leq t$, où p est l'arité de s , il existe $s' \in R_{\alpha(G)}$ avec $\gamma(s') = \Pi(c_1), \dots, \Pi(c_p)$;
 4. $type(r) \geq type(s)$ et il existe dans $R_{\alpha(G)}$ toutes les sommets relation s' tels que $type(s) \leq type(s')$, donc il existe en particulier un sommet relation r' tel que $type(r) = type(r')$ avec $\gamma(r') = \Pi(c_1), \dots, \Pi(c_p)$.

Π' est donc bien une $\alpha(\mathcal{S})$ -projection de H dans $\alpha(G)$.

(\Leftarrow) Il nous reste maintenant à montrer que si Π' est une $\alpha(\mathcal{S})$ -projection de H dans $\alpha(G)$, alors $\Pi = \alpha^{-1} \circ \Pi'$ est une \mathcal{S} -projection de H dans G .

- Π est bien une application de C_H dans C_G
- Montrons que $\forall c \in C_H, l_H(c) \geq l_G(\Pi(c))$:

1. soit $c \in C_H$;
2. Π' étant une projection, $\exists \Pi'(c) \in C_{\alpha(G)}$ tel que $type(c) = type(\Pi'(c))$ (car le T_C ne contient que \top) et $marker(c) \geq marker(\Pi'(c))$;
3. $\Pi'(c) \in C_{\alpha(G)}$, donc $\exists ! \alpha^{-1} \circ \Pi'(c) \in C_G$ tel que $type(\Pi'(c)) = type(\alpha^{-1} \circ \Pi'(c))$ et $marker(\Pi'(c)) = marker(\alpha^{-1} \circ \Pi'(c))$;
4. comme $\alpha^{-1} \circ \Pi'(c) = \Pi(c)$, il vient que $l_H(c) \geq l_G(\Pi(c))$.

- Montrons maintenant que $\forall r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$, il existe $r' \in R_G$ avec $\gamma(r') = (\Pi(c_1), \dots, \Pi(c_p))$ et $l_H(r) \geq l_G(r')$.

1. soit $r \in R_H$ avec $\gamma(r) = (c_1, \dots, c_p)$;
2. Π' est une $\alpha(\mathcal{S})$ -projection, donc $\exists s \in R_{\alpha(G)}$ tel que $type(r) = type(s)$ et $\gamma(s) = \Pi'(c_1), \dots, \Pi'(c_p)$;
3. $s \in R_{\alpha(G)}$ alors, par construction, $\exists ! r' \in R_G$ tel que $r' = \alpha^{-1}(s)$, $type(s) \geq type(r')$ et $\gamma(r') = \alpha^{-1} \circ \Pi'(c_1), \dots, \alpha^{-1} \circ \Pi'(c_p)$;
4. or $\Pi = \alpha^{-1} \circ \Pi'$, donc $\gamma(r') = (\Pi(c_1), \dots, \Pi(c_p))$ et $type(r) \geq type(r')$.

Nous en concluons que Π est une \mathcal{S} -projection de H dans G . □

Théorème 5 *Il existe une transformation polynomiale τ telle que :*

1. τ associe à tout graphe conceptuel g défini sur \mathcal{S} un graphe conceptuel $\tau(g)$ défini sur $\tau(\mathcal{S})$
2. τ associe à tout support \mathcal{S} un support plat purement relationnel $\tau(\mathcal{S})$
3. $H \succeq_{\mathcal{S}} G$ ssi $\tau(H) \succeq_{\tau(\mathcal{S})} \tau(G)$

Preuve Posons $\tau = \alpha \circ \delta$. Ces deux fonctions sont définies. Nous avons déjà vu que, d'après le théorème 4, que $H \succeq_{\mathcal{S}} G \Leftrightarrow H \succeq_{\alpha(\mathcal{S})} \alpha(G)$ et que $H \succeq_{\mathcal{S}} G \Leftrightarrow \delta(H) \succeq_{\delta(\mathcal{S})} \delta(G)$ d'après le théorème 3. Nous pouvons en déduire que

$$H \succeq_{\mathcal{S}} G \Leftrightarrow \delta(H) \succeq_{\alpha \circ \delta(\mathcal{S})} \alpha \circ \delta(G)$$

Or, d'après la proposition 2 :

$$\delta(H) \succeq_{\alpha \circ \delta(\mathcal{S})} \alpha \circ \delta(G) \Leftrightarrow \alpha \circ \delta(H) \succeq_{\alpha \circ \alpha \circ \delta(\mathcal{S})} \alpha \circ \alpha \circ \delta(G)$$

En utilisant la propriété 5, nous obtenons

$$\delta(H) \succeq_{\alpha \circ \delta(\mathcal{S})} \alpha \circ \delta(G) \Leftrightarrow \alpha \circ \delta(H) \succeq_{\alpha \circ \delta(\mathcal{S})} \alpha \circ \delta(G)$$

Finalement.

$$H \succeq_{\mathcal{S}} G \Leftrightarrow \alpha \circ \delta(H) \succeq_{\alpha \circ \delta(\mathcal{S})} \alpha \circ \delta(G)$$

En remplaçant $\alpha \circ \delta$ par τ nous obtenons la transformation souhaitée. \square

2.4 Équivalence entre $\text{BD} + \mathcal{S}$ et BD

Récapitulons : nous avons montré l'équivalence entre graphes conceptuels (avec support) et bases de données avec support, et ensuite entre graphes conceptuels (avec support) et graphes conceptuels sans support (ou plus exactement définis sur un support purement relationnel plat). Grâce à ces deux résultats, il nous est maintenant possible de montrer facilement l'équivalence entre graphes conceptuels et bases de données sans support, et finalement, entre bases de données avec support et bases de données sans support

Corollaire 1 *Soient G et H deux graphes conceptuels définis sur un support \mathcal{S} . Alors :*

$$H \succeq_{\mathcal{S}} G \Leftrightarrow \text{il existe une réponse à } q \circ \delta(H) \text{ dans } b \circ \alpha \circ \delta(G)$$

Preuve Le théorème 5 nous permet d'affirmer que « il existe une réponse à $q \circ \delta(H)$ dans $b \circ \alpha \circ \delta(G)$ » équivaut à « il existe une réponse à $q \circ \tau(H)$ dans $b \circ \tau(G)$ ».

Si G et H sont des GC tels qu'il existe une \mathcal{S} -projection de H dans G , alors il est possible de construire $\tau(H)$ et $\tau(G)$, définis sur $\tau(\mathcal{S})$, tels qu'il existe une $\tau(\mathcal{S})$ -projection de $\tau(H)$ dans $\tau(G)$. De plus, $\tau(\mathcal{S})$ est un support plat.

Nous savons aussi, d'après le théorème 2, qu'il est possible de créer la requête et la base de données associés à H et G , respectivement. Il nous suffit alors de remarquer que, si $\tau(H)$ et $\tau(G)$ sont définis sur un support plat, les transformations q et b engendrent, respectivement, une requête sans support et une base de données sans support.

Il en découle que si \mathcal{S} est un support plat, alors une \mathcal{S} -réponse équivaut à une réponse. \square

Corollaire 2 *Soient Q une requête conjonctive et B une base de données définie sur un support \mathcal{S} . Il est possible de définir une paire de transformations τ'_1 et τ'_2 telle qu'il existe une \mathcal{S} -réponse à Q dans B ssi il existe une réponse à $\tau'_1(Q)$ dans $\tau'_2(B)$.*

$$\begin{array}{ccc}
\text{GC} + \mathcal{S} & \xrightarrow{\text{thm. 2}} & \text{BD} + \mathcal{S} \\
\text{thm. 5} \downarrow & & \\
\text{GC} & \xrightarrow{\text{thm. 2}} & \text{BD}
\end{array}$$

FIG. 2.6 – Schéma d'équivalence entre les 4 modèles

Preuve Soient Q une requête conjonctive et B une base de données définie sur un support \mathcal{S} . Les transformations q et b étant bijectives (propriétés 1 et 1), nous pouvons dire que (théorème 2) :

$$\text{il existe une } \mathcal{S}\text{-réponse à } Q \text{ dans } B \Leftrightarrow q^{-1}(Q) \succeq_{\mathcal{S}} b^{-1}(B)$$

Le corollaire 1 nous permet de déduire que

$$q^{-1}(Q) \succeq_{\mathcal{S}} b^{-1}(B) \Leftrightarrow \text{il existe une réponse à } q \circ \tau \circ q^{-1}(Q) \text{ dans } b \circ \tau \circ b^{-1}(B)$$

En posant $\tau'_1 = q \circ \tau \circ q^{-1}$ et $\tau'_2 = b \circ \tau \circ b^{-1}$, nous arrivons au résultat attendu. \square

Chapitre 3

Résultats expérimentaux

3.1 Générateur de graphes

Nous avons utilisé un générateur de graphes aléatoires qui est l'adaptation en graphes conceptuels du générateur aléatoire de CSP (*Constraint satisfaction problem*) présenté dans l'article de Ke Xu et Wei Li ([XL00]).

À chaque appel du générateur, celui-ci crée un support et deux graphes, associés à ce support. Le premier graphe créé, que nous appellerons H , fera office de requête, et le deuxième nous servira de base de faits (nous le noterons G). Examinons, tout d'abord, les paramètres nécessaires à la génération. Ils sont au nombre de cinq :

n : nombre de sommets concept dans H

a : sert au calcul du nombre de sommets concept dans G

r : sert au calcul du nombre de sommets relation dans H

p : « dureté »

k : arité des relations

Regardons maintenant, plus en détail, comment, à partir des paramètres reçus, le générateur crée le support et les graphes (*nota bene* : le support est généré au fur et à mesure de la construction des graphes ; les relations, quant à elles, sont toutes du même type, aussi bien dans H que dans G).

Algorithme 1 : Générateur de graphes

Données : n, a, r, k

Résultat : un support et deux graphes H et G

$d \leftarrow n^a$;

créer un nouveau support;

ajouter au support le type de concept \top ;

ajouter au support le type de relation \top_k ;

// Création des sommets concept de H

pour $i \leftarrow 1$ à n **faire**

┌ ajouter au support un type de concept tc plus spécifique que \top ;

└ ajouter au graphe H un sommet concept de type tc ;

// Création des sommets concept de G

pour chaque *sommet concept* c de H **faire**

┌ $ct \leftarrow$ type de c ;

└ // nous appellerons les sommets créés lors cette étape
les sommets dérivés de c

pour $i \leftarrow 1$ à d **faire**

┌ ajouter au support un type de concept tc' plus spécifique que

tc ;

└ ajouter au graphe G un sommet concept de type tc' ;

// Création des sommets relation dans H

pour $i \leftarrow 1$ à $r * n * \log(n)$ **faire**

┌ choisir k sommets concept distincts dans H , tels qu'il n'existe
aucun sommet relation dans H défini sur ces sommets, dans
l'ordre;

└ ajouter au graphe H un sommet relation ayant pour voisins les
sommets choisis dans l'étape précédente;

// Création des sommets relation dans G

// Pour chaque sommet relation de H , $(1-p) * d^k$ sommets
concepts seront créés dans G

pour chaque *sommet relation* r de H **faire**

┌ **pour** $i \leftarrow 1$ à $(1-p) * d^k$ **faire**

└ $l \leftarrow$ voisins de r $l' \leftarrow$ liste vide **pour chaque** *sommet concept*
 c de l **faire**

└└ choisir un sommet concept c' dans G tel que c' soit dérivé
de c ;

└└ s'il existe dans G une relation dont la liste des voisins est égale
à l' , recommencer la création de l' ;

└└ ajouter au graphe G un sommet relation de type \top_k ayant pour
voisins les sommets concept de la liste l' ;

Un des paramètres les plus importants passé au générateur est p , la *dureté*, qui permet de déterminer le nombre de sommets relation qui seront créés dans le graphe G . Si p vaut (ou est proche de) 0, G contiendra beaucoup de relations et il sera très facile de projeter H dans G . À l'inverse, plus p s'approche de 1, moins il y a de relations dans G , et il devient alors de plus en plus difficile — voire impossible si p vaut 1 — de trouver une projection de H dans G . Entre ces deux extrêmes, il existe une valeur de p pour laquelle le graphe G contiendra exactement une projection de H , et c'est ce point que nous observons la *transition de phase*.

Nous nous intéressons, dans le cadre de nos tests, à un problème de satisfiabilité : nous ne considérerons que le temps nécessaire pour trouver la première solution. Dans ce cadre, il apparaît que si p est proche de 0, nous obtiendrons une réponse rapidement. En effet, beaucoup de chemins dans l'exploration de l'arbre des solutions constitueront une solution. De la même manière, quand p est proche de 1, nous obtenons aussi très vite une réponse, car nous voyons rapidement qu'il n'y a pas de solution. Le point critique est le cas où il n'existe qu'une seule solution et, dans cette situation, il peut être nécessaire d'explorer toute l'arbre de recherche avant de trouver la solution.

Nous nous attendions donc à obtenir une courbe en forme de cloche, et cette expectative a été confirmée par les tests.

Protocole pour les tests : pour chaque séance de tests, nous avons généré des graphes en fixant les variables n , a , r et k et en faisant varier p . Ainsi, pour chaque p (qui a pris des valeurs entre 0 et 1 par pas de 0,05¹), 100 couples de graphes avec leur support furent générés.

À partir des graphes créés par le générateur, nous effectuons les transformations présentées dans le chapitre précédent pour obtenir des graphes réifiés sur support réifié, réifiés expansés sur support réifié aplati, requête et base de données avec support et, finalement, requête et base de données sans support.

3.2 Tests

Nous avons fait des tests avec CoGITaNT ([GS98]) pour les graphes conceptuels, et Oracle pour les bases de données sans support, un SGBD réputé pour son efficacité.

¹Un seul test a été effectué avec un pas de 0,02, celui dont le résultat est présenté dans la courbe de la figure 3.1. Pour les autres, nous avons gardé la valeur 0,02 pour que les tests soient un peu plus rapides.

Les tests avec CoGITaNT se sont bien déroulés. La figure 3.1 illustre le résultat d'un test sur des graphes générés avec les paramètres $n = 15, a = 0.7, r = 3$ et $k = 2$. Dans cette courbe, nous pouvons observer très nettement la transition de phase, qui se situe ici aux alentours de $p = 0,1$. À la transition, CoGITaNT met, en moyenne, environ 1000 ms pour trouver une projection. Ce résultat correspond à ce que nous attendions (voir section précédente).

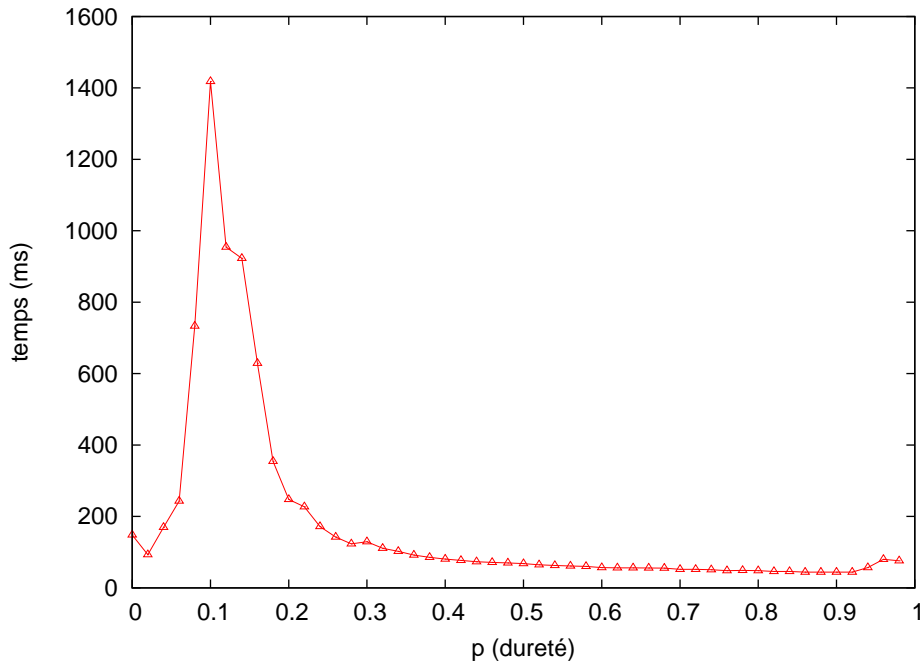


FIG. 3.1 – Résultats des tests ($n = 15, a = 0.7, r = 3, k = 2$)

Malheureusement, les tests ne se sont pas bien déroulés avec Oracle. Après avoir transformé un GC H en une requête SQL et un GC G réifié et expansé en une BD, nous avons essayé d'exécuter la requête sous Oracle. Ce fut un échec total, car les temps de réponse étaient gigantesques². Si le premier réflexe a été de mettre en cause les divers algorithmes de transformation, cette hypothèse fut vite écartée après des tests et contrôles poussés sur les algorithmes.

² *Vraiment* gigantesques. Sur des exemples avec $n=6$, là où CoGITaNT ou même notre propre algorithme ne mettent que quelques millisecondes, Oracle met une vingtaine de minutes! En extrapolant, un test qui demanderait une trentaine de secondes à réaliser avec cogitant nécessiterait, avec Oracle, un peu plus de trois mois.

Cependant, même si le résultat mettait du temps à venir, celui-ci était correct. Le problème venait donc d'Oracle lui-même. En analysant ce que faisait Oracle pour chercher et renvoyer les résultats, quelques grands problèmes sont apparus : tout d'abord, nous n'avons pas réussi à limiter le nombre de résultats d'une requête SQL. Et comme nous l'avons précisé dans la section précédente, nous voulons ici le temps nécessaire pour trouver la *première* solution.

À lui seul, ce fait pourrait expliquer la lenteur des réponses Oracle : dans des graphes avec n grand et p petit, il y a un très grand nombre de résultats, et Oracle les stockait tous pour les renvoyer à la fin de la recherche³, ce qui consomme une très grande quantité de mémoire, obligeant même la machine à *swapper*⁴ ce qui, dès lors, empêche de mesurer correctement le temps.

Un autre problème est dû au fonctionnement même des requêtes SQL : le SGBD fait une jointure des tables du **FROM**, des sélections selon les conditions du **WHERE** et une projection sur les colonnes du **SELECT**, de manière plus ou moins optimisée suivant le SGBD. Dans nos tests, le champ **WHERE** des requêtes générées ne contient jamais quelque chose comme « où la deuxième colonne de la table t vaut A », mais plutôt des conditions comme « où la deuxième colonne de la table t a la même valeur que la première colonne de la table u », ce qui empêche le SGBD de faire une sélection dans les lignes des tables avant la jointure. En plus, les tables manipulées sont plutôt grandes⁵, ce qui résulte en une table gigantesque, une fois le produit cartésien calculé.

Finalement, il est important de préciser que Oracle travaille sur la transformation en base de données d'un graphe conceptuel *réifié et expansé*. La réification est une transformation qui alourdit beaucoup le graphe. On met environ cent fois plus de temps pour trouver une projection dans un graphe réifié que dans un graphe non réifié. L'expansion entraîne aussi un surcoût, mais celui-ci est négligeable⁶.

³Il faut noter qu'avec CoGITaNT l'on retrouve la même situation : en demandant tous les résultats dans des projections qui retournent un très grand nombre de résultats, la consommation de mémoire pour stocker les différents résultats peut s'élever au point de faire planter la machine.

⁴Se servir du fichier d'échange, qui est un fichier d'un type particulier, permettant d'utiliser une partie d'un disque dur exactement de la même manière que de la mémoire vive, étendant ainsi cette dernière. C'est donc de la mémoire virtuelle. Comme la mémoire coûte bien plus cher que l'espace sur un disque dur, c'est rentable, mais le disque est bien plus lent que la mémoire électronique (dans un rapport de 1 à 100.000). Source : Dictionnaire d'informatique francophone, <http://www.linux-france.org/prj/jargonf>

⁵*Relativement* grandes. Pour une « vrai » base de données, elles ne le sont pas vraiment, mais c'est le fait qu'on fasse des jointures en utilisant plusieurs fois la même table qui crée des très grandes tables.

⁶Cf. annexe sur les transformations p. 35

Face à ces problèmes, nous avons décidé de procéder d'une autre manière.

3.3 Tests - deuxième essai

Pour pallier aux problèmes rencontrés avec Oracle, nous avons choisi une approche très différente de la première : nous avons développé nos propres algorithmes de projection, qui consistent en un algorithme simple de back-track qui pose des questions élémentaires à une base de faits, celle-ci pouvant être soit un graphe conceptuel, soit une base de données (dans ce cas, des requêtes *élémentaires*⁷ sont faites au SGBD). Cette approche nous permet de comparer la projection dans un graphe conceptuel avec support avec celle dans une base de données avec support.

3.3.1 Exemple

Voyons comment cet algorithme travaille dans le cas d'une projection sur une base de données. Soit la situation présentée dans la figure 3.2, dans laquelle sont présents un support, deux graphes définis sur ce support, H et G , le premier étant le « graphe question » et le deuxième étant le « graphe base de faits ». Est aussi représentée une base de données, définie elle aussi sur le support de la figure, construite à partir du graphe G à l'aide des transformations présentées dans le chapitre précédent.

La projection de H dans G se fait de manière classique. La projection de H dans $BD(G)$, quant à elle, mérite un peu plus d'attention.

Supposons que l'on commence par projeter le sommet $c0$. Les requêtes SQL associées à cette projection sont (les valeurs retournées par la requête sont indiquées après les flèches) :

```
SELECT T.marker FROM T
SELECT v.marker FROM v -> B
SELECT m.marker FROM m -> C
SELECT b.marker FROM b -> A
SELECT c.marker FROM c -> D
```

Une des valeurs retournées est choisie (supposons que ce soit A) et on passe à la projection de $c1$.

⁷Plutôt que de faire une seule grosse requête, obtenue à partir du « graphe question », nous faisons plusieurs petites requêtes, comme « où peut-on envoyer le sommet concept x ? ».

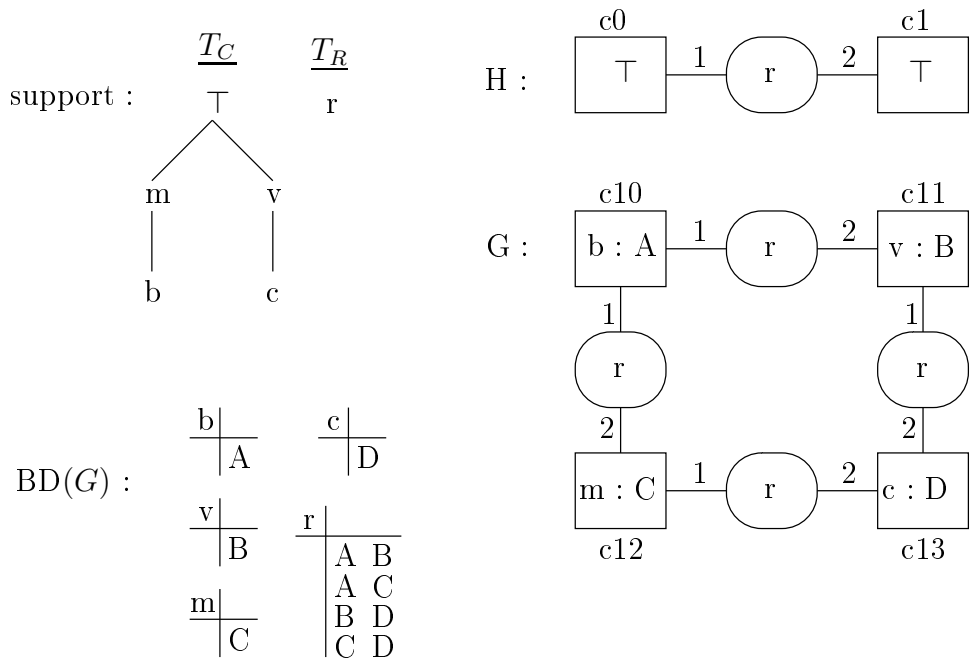


FIG. 3.2 – Un support, un graphe question H , un graphe fait G et la transformation de G en $BD + \mathcal{S}$

Une fois qu'un sommet a été affecté, la situation n'est plus la même, car $c1$ est relié à une relation qui possède des voisins déjà projetés ($c0$), il faut donc tenir compte de cette contrainte. Les requêtes pour la projection de $c1$ sont :

```

SELECT T.marker FROM T, r WHERE r.v1 = 'A' AND r.v2 = T.marker
SELECT v.marker FROM v, r WHERE r.v1 = 'A' AND r.v2 = v.marker -> B
SELECT m.marker FROM m, r WHERE r.v1 = 'A' AND r.v2 = m.marker -> C
SELECT b.marker FROM b, r WHERE r.v1 = 'A' AND r.v2 = b.marker
SELECT c.marker FROM c, r WHERE r.v1 = 'A' AND r.v2 = c.marker

```

Nous obtenons déjà deux affectations possibles pour le couple $[c0, c1]$: $[A, B]$ et $[A, C]$. En faisant le backtrack pour choisir d'autres valeurs pour $c0$, nous obtenons deux autres solutions, $[B, D]$ et $[C, D]$.

Ce petit exemple illustre bien la grande quantité de requêtes nécessaires. Si le type de relation r avait eu un descendant, s par exemple, le nombre de requêtes aurait été multiplié par deux.

3.3.2 Résultats

Cette fois les tests ont pu être menés à bien, même si les résultats ne sont pas encourageants. Les graphes utilisés ici ont été créés avec les paramètres $n = 10, a = 0.7, r = 3$ et $k = 2$. La figure 3.3⁸ représente le résultat obtenu avec notre algorithme de backtrack sur un graphe conceptuel. Le résultat du backtrack sur une base de données est décrit dans la figure 3.4.

Nous constatons que, malgré le fait que le test se soit bien déroulé et que les résultats soient corrects, le temps de réponse est beaucoup trop lent. Au point de la transition de phase, il faut environ mille fois plus de temps pour obtenir une réponse en interrogeant une base de données qu'il ne faut pour projeter un graphe conceptuel dans un autre (55,8 ms pour GC contre 56751 pour BD).

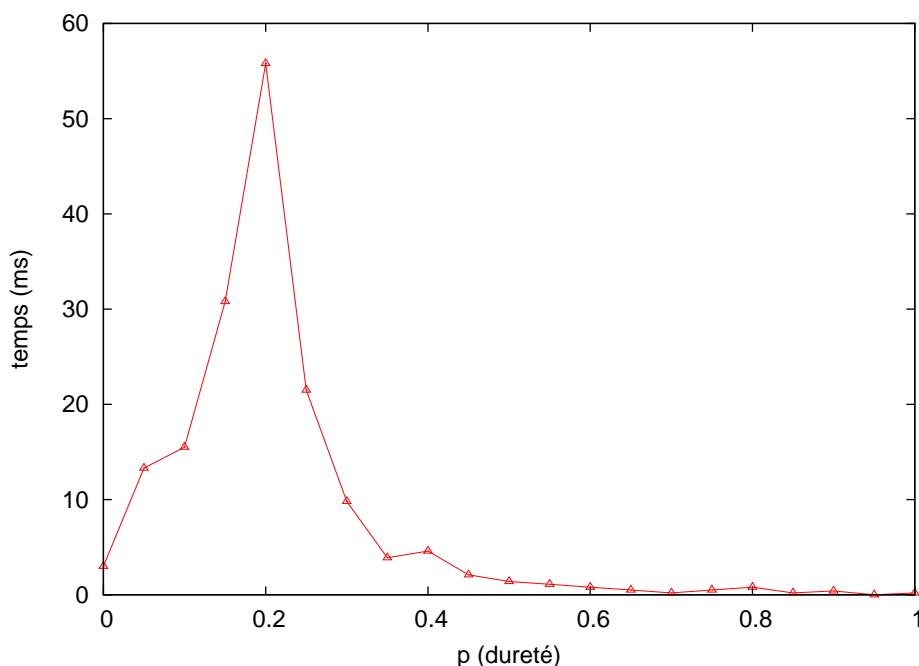


FIG. 3.3 – Backtrack sur un graphe conceptuel

⁸Le lecteur attentif aura remarqué que, apparemment, les résultats obtenus avec notre algorithme sont meilleurs que ceux obtenus avec CoGITaNT (cf. figure A.1 p. 36). Cela s'explique simplement par le fait que, contrairement à CoGITaNT, qui projette sommets concepts et sommets relation, nous ne projetons que les sommets concept, ce qui nous favorise au moins dans les graphes de taille modeste. Dès que la taille du graphe commence à être un peu plus importante, CoGITaNT prend rapidement le dessus.

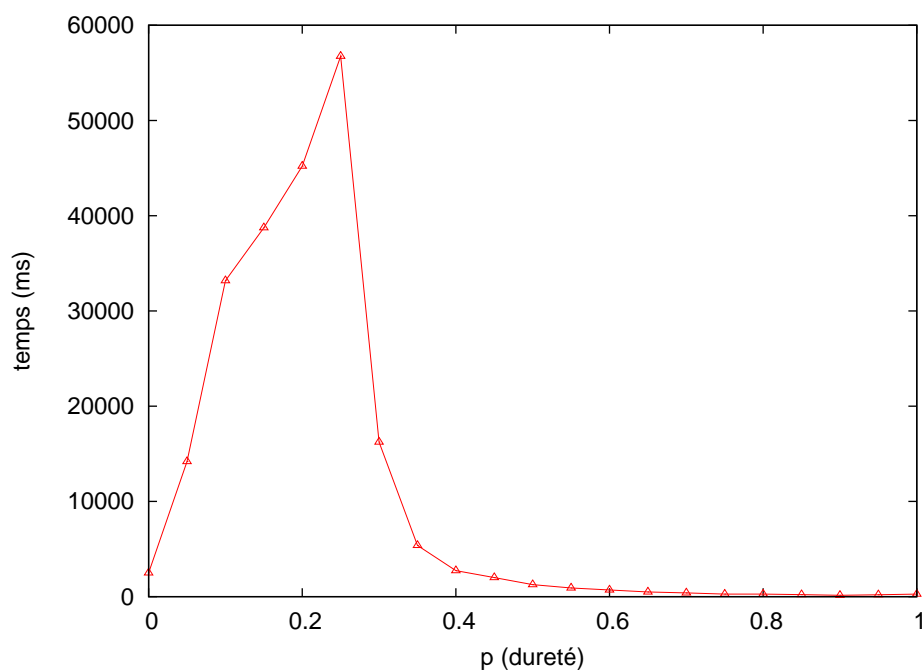


FIG. 3.4 – Backtrack sur une base de données

3.4 Conclusion

Fâcheusement, nous devons conclure que le travail que nous avons effectué ces derniers mois, en dépit d'être réalisable en théorie, ne nous semble pas être utilisable en pratique.

Cependant, un travail ultérieur pourrait être fait pour essayer de déterminer pour quelles raisons les résultats sont si mauvais avec les bases de données. Un chemin à explorer serait alors d'examiner différents « types » de graphes, comme des graphes très denses ou des graphes, au contraire, très peu denses ; pourraient aussi être comparés les graphes qui engendrent des bases de données avec un grand nombre de tables peu peuplées avec ceux dont la transformation en BD résulte en une petite quantité de tables, mais qui contiendraient un très grand nombre de tuples.

Annexe A

Influence des transformations dans le temps de projection

La réification est une opération très « coûteuse », elle augmente considérablement le temps nécessaire pour trouver une projection, surtout lors de la transition de phase.

Les trois figures suivantes résument les résultats des tests effectués avec des graphes sans aucune transformation (première figure), des graphes réifiés (deuxième figure) et des graphes réifiés et expansés (dernière figure). Les paramètres passés au générateur de graphes pour ces tests furent $n = 10$, $a = 0,7$, $r = 3$ et $k = 2$. Les temps sont présents dans le tableau ci-dessous.

Transformation	temps (ms)
aucune	134
réification	13 122
réification et expansion	13 294

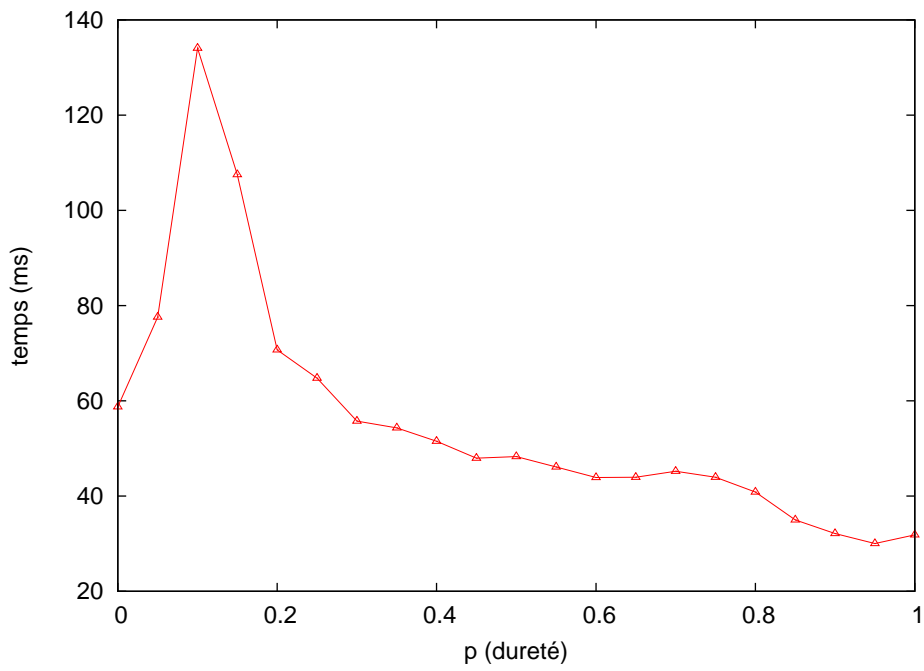


FIG. A.1 – Aucune transformation

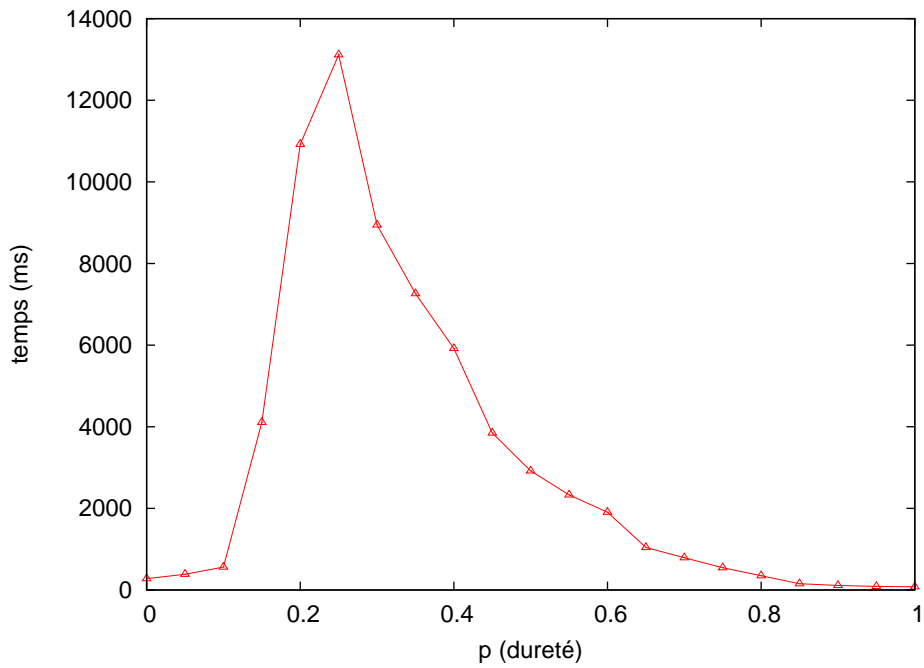


FIG. A.2 – Graphes réifiés

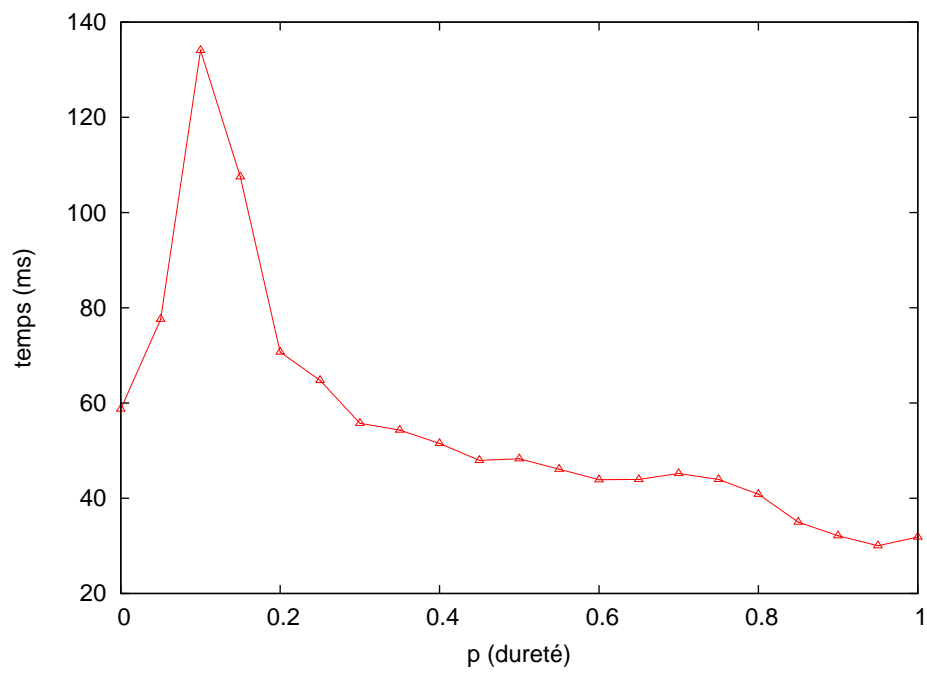


FIG. A.3 – Graphes réifiés et expansés

Bibliographie

- [CM92] Michel Chein and Marie-Laure Mugnier. Conceptual graphs : fundamental notions. *Revue d'Intelligence Artificielle*, 6 :365–406, 1992.
- [GS98] David Genest and Eric Salvat. A platform allowing typed nested graphs : How cogito became cogitant. In Tools Conceptual Structures : Theory and proceedings of the sixth International Conference on Conceptual Structures (ICCS '98) Applications, editors, *Lecture Notes in Artificial Intelligence*, volume 1453, pages pages 154–161. Springer, 1998.
- [Sow76] John F. Sowa. Conceptual graphs for a database interface. *IBM Journal of Research and Development*, vol. 20(no. 4) :336–357, 1976.
- [Sow84] John F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1984.
- [XL00] Ke Xu and Wei Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12 :93–103, 2000.