Université de Montpellier - Master informatique Janvier 2024

Théorie des bases de données et de connaissances (HAI933I)

Examen

Durée totale : 2 heures
Document autorisé : 1 feuille A4 manuscrite recto-verso

Toutes vos réponses doivent être justifiées. Une affirmation sans justification ne sera pas prise en compte. Le barème est donné à titre indicatif et peut varier légèrement.

1 Théorie des bases de données (4 points)

Question 1. Considérez la base de données suivante :

Films

Titre	Réalisateur	Acteur
The Imitation Game	Tyldum	Cumberbatch
The Imitation Game	Tyldum	Knightley
...	...	...
Internet's Own Boy	Knappen	Swartz
Internet's Own Boy	Knappen	Lessig
Internet's Own Boy	Knappen	Berners-Lee
...	...	...
Dogma	Smith	Damon
Dogma	Smith	Affleck

Cinémas

Cinéma	Adresse	Téléphone
UFA	St. Peters Str. 24	4825825
Schauburg	Königs Str. 55	8032185
...	...	...

Programme

Cinéma	Titre	Heure
Schauburg	The Imitation Game	19:30
Schauburg	Dogma	20:45
UFA	The Imitation Game	22:45

Écrivez les requêtes suivantes en tant que "first-order queries" pour cette base de données :

1. Écrivez une requête booléenne pour déterminer s'il existe un film qui a été réalisé par deux réalisateurs différents.
2. Écrire une requête pour lister tous les cinémas qui ne projettent pas de film réalisé par "Tyldum".

Question 2. Cette requête est-elle acyclique ? Expliquez pourquoi.

$$\exists z_1, \ldots, z_6 . P(z_3, z_1, z_2) \wedge S(z_1, z_3, z_5) \wedge R(z_4, z_3, z_2) \wedge V(z_1, z_6, z_2)$$

Si cette requête est acyclique, dessinez son "join tree".

2 Règles existentielles (5 points)

Question 1. Qu'est-ce qu'un modèle universel d'une base de connaissances ? Quel est son intérêt pour déterminer si une requête conjonctive est conséquence logique d'une base de connaissances ?

Question 2. Soit la base de connaissances $\mathcal{K} = (F, \mathcal{R})$ où $F = \{p(a, b)\}$ et $\mathcal{R} = \{R_1, R_2, R_3\}$ où :

$R_1 : p(x, y) \rightarrow q(y)$
$R_2 : p(x, y) \rightarrow \exists z \; p(y, z)$
$R_3 : q(x) \rightarrow p(x, x)$

On effectue le *restricted chase* sur $\mathcal{K}$ et on procède *en largeur*[1].

2.1 Détaillez les 3 premières étapes de largeur : donnez les nouveaux déclencheurs, et pour chacun déterminez s'il est actif, et si oui donnez les atomes produits.

2.2 Donnez le résultat du restricted chase en largeur sur $\mathcal{K}$.

[1] À chaque étape i : (1) on cherche tous les déclencheurs (triggers) (R, h), où h est un nouvel homomorphisme du corps de la règle R dans la base de faits F_{i-1}; (2) pour chacun de ces déclencheurs pris dans un ordre quelconque, on teste s'il est actif selon le critère du restricted chase sur la base de faits courante, et s'il l'est on effectue l'application associée en ajoutant les atomes produits à la base de faits courante.

Question 3. La base de connaissances $\mathcal{K}$ possède-t-elle un modèle universel fini ? Justifiez votre réponse.

Question 4. L'ensemble de règles $\mathcal{R}$ est-il aGRD (graphe de dépendances des règles acyclique)? Est-il weakly-acyclic (graphe des positions sans circuit dangereux) ? Quand votre réponse est non on ne vous demande pas de donner le graphe associé complet, mais seulement les arcs qui justifient votre réponse.

Question 5. Etant donné un ensemble de règles $\mathcal{R}$, on dit qu'une règle $R \in \mathcal{R}$ est redondante si $\mathcal{R}$ et $\mathcal{R} \setminus \{R\}$ sont logiquement équivalents. Pour toute base de connaissances $(F, \mathcal{R})$, on peut donc supprimer R de $\mathcal{R}$ en gardant une base de connaissances équivalente. A votre avis, l'ensemble $\mathcal{R}$ de la question 2 contient-il une règle redondante ? Expliquez.

Question 6 (Bonus - plus difficile) : Proposez une méthode pour tester si une règle est redondante.

3 Requêtes conjonctives avec négation (4 points)

On considère des requêtes conjonctives avec négation. On impose que la négation soit sûre : toute variable apparaissant dans un littéral négatif apparaît aussi dans un littéral positif.

Question 1. Soit la requête booléenne :

$$Q = \exists X \exists Y_1 \exists Y_2 \; p(X, Y_1) \wedge p(X, Y_2) \wedge r(Y_1, Y_2) \wedge \neg q(Y_1) \wedge q(Y_2)$$

On fait l'hypothèse du monde clos. Soit la base de données :

$$D = \{p(a, b), p(a, c), p(a, d), q(c), r(b, c), r(c, b), r(c, d), r(d, c)\}$$

D répond-elle positivement à Q ? Pourquoi ?

Question 2. Soit la requête :

$$Q'(Y_1, Y_2) = \exists X \; p(X, Y_1) \wedge p(X, Y_2) \wedge r(Y_1, Y_2) \wedge \neg q(Y_1) \wedge q(Y_2)$$

Remarquez que Q' ne diffère de Q que par ses variables réponses. Donnez l'ensemble des réponses à Q' sur D.

Question 3. Supposons maintenant que l'on fasse l'hypothèse du monde ouvert. Pourquoi D répond-elle négativement à Q ?

Question 4. Toujours dans le cadre du monde ouvert, soit F (obtenue à partir de D en remplaçant $q(c)$ par $q(b)$ et $\neg q(d)$) :

$$F = \{p(a, b), p(a, c), p(a, d), q(b), \neg q(d), r(b, c), r(c, b), r(c, d), r(d, c)\}$$

F répond-elle positivement à Q? Pourquoi ?

Question 5. Donnez l'ensemble des réponses à Q' sur F.

Question 6. Donner l'ensemble des réponses à :

$$Q''(X) = \exists Y_1 \exists Y_2 \; p(X, Y_1) \wedge p(X, Y_2) \wedge r(Y_1, Y_2) \wedge \neg q(Y_1) \wedge q(Y_2)$$

sur F. Remarquez que Q'' ne diffère de Q' que par les variables réponses.

4 Modèles stables (7 points)

L'objectif de cette partie de l'examen est de tester la satisfiabilité d'une formule de 2QBF en utilisant les modèles stables (vous utiliserez la syntaxe des règles existentielles avec négation (REN) vue en cours : en particulier, vous n'aurez pas droit à la disjonction en tête de règle).

Remarque préliminaire: *Les questions 1 à 7 ne nécessitent pas de comprendre 2QBF pour y répondre. Je vous conseille cependant fortement de lire la section 4.1 (introduction à 2QBF) pour comprendre là où les questions doivent vous mener.*

Rappels de logique des propositions: une formule F de logique des propositions est *satisfiable* si il existe une valuation σ de ses variables (ou symboles propositionnels) telle que $\sigma(F)$ s'évalue à `true`. Elle est *valide* si, pour toute valuation σ de ses variables, $\sigma(F)$ s'évalue à `true`.

4.1 Une introduction à 2QBF

Une formule de 2QBF est de la forme $\exists\vec{x}\forall\vec{y}\,F(\vec{x},\vec{y})$ où $F(\vec{x},\vec{y})$ est une formule propositionnelle dont les variables (ou symboles propositionnels) sont celles de $\vec{x}\cup\vec{y}$. Une telle formule est satisfiable ssi il existe une valuation des variables de $\vec{x}$ dans {`true`, `false`} telle que pour toute valuation des variables de $\vec{y}$, la valeur de vérité de la formule $F(\vec{x},\vec{y})$ est `true`. En d'autres termes, cette formule est satisfiable ssi il existe une valuation σ des variables de $\vec{x}$ dans {`true`, `false`} telle que la formule $\sigma(F(\vec{x},\vec{y}))$ est valide.

Sans perte de généralité, nous considérons par la suite que la formule $F(\vec{x},\vec{y})$ est une 3DNF, c'est à dire une disjonction de conjonctions contenant chacune au plus 3 littéraux (variable x ou sa négation).

Exemples: La formule 2QBF $F = \exists x\exists y\forall z\,F'$ avec $F' = (x\wedge\neg y\wedge z)\vee(x\wedge\neg z)$ se lit "il existe des valeurs booléennes x et y telles que, pour toute valeur booléenne z, la formule F' est vraie". Soit la valuation $\sigma = \{x : \texttt{true}, y : \texttt{false}\}$. On a $\sigma(F') = (\texttt{true}\wedge\texttt{true}\wedge z)\vee(\texttt{true}\wedge\neg z)$, qui est équivalente à $z\vee\neg z$ qui est valide. La formule F est donc satisfiable.

Soit la formule 2QBF $G = \exists x\forall y\,G'$ avec $G' = (x\wedge y)\vee(\neg x\wedge\neg y)$. Considérons les valuations $\sigma_1 = \{x : \texttt{true}\}$ et $\sigma_2 = \{x : \texttt{false}\}$. On a $\sigma_1(G') = (\texttt{true}\wedge y)\vee(\texttt{false}\wedge\neg y)$ qui est équivalente à $y\vee\texttt{false}$, équivalente à y, qui est invalide (c'est-à-dire non valide). Nous avons également $\sigma_2(G') = (\texttt{false}\wedge y)\vee(\texttt{true}\wedge\neg y)$ qui est équivalente à $\texttt{false}\vee\neg y$, équivalente à $\neg y$, également invalide. Comme aucune substitution par une valuation de x ne produit de formule valide, alors la formule G est insatisfiable.

Au cours de cet exercice, vous devrez étudier (sans l'écrire explicitement) un *traducteur* qui, à partir d'une *formule* 2QBF $Q = \exists\vec{x}\forall\vec{y}\,F(\vec{x},\vec{y})$, génère un *programme* $\Pi(Q)$ écrit avec des règles existentielles avec négation (REN). Ce traducteur devra respecter la propriété suivante: la valuation $\sigma = \{x_1 : b_1,\ldots,x_k : b_k\}$ (où les x_i sont les variables quantifiées existentiellement de Q et les b_i sont les booléens `true` ou `false`) est telle que $\sigma(F(\vec{x},\vec{y}))$ est valide si et seulement si il existe un modèle stable de $\Pi(Q)$ dont les prédicats ayant pour nom de prédicat *val* sont exactement $val(x_1,b_1),\ldots,val(x_k,b_k)$. Ainsi, la formule Q sera satisfiable ssi le programme $\Pi(Q)$ admet un modèle stable.

4.2 Valuation des variables existentielles

Cette partie du travail vise à générer toutes les valuations possibles pour les variables quantifiées existentiellement d'une 2QBF.

Question 1 Écrire un programme Π^1 dont les modèles stables contiennent toutes les valuations possibles pour les variables propositionnelles x_1 et x_2. Pour chacune de ces deux variables (et ici nous considérons x_1), chaque modèle stable devra contenir soit l'atome $val(x_1,\texttt{true})$, soit l'atome $val(x_1,\texttt{false})$, mais pas les deux. Remarquons au passage que les variables propositionnelles de 2QBF écrites avec des minuscules deviennent des constantes en REN.

Question 2 Remarquons que le programme Π^1 est un programme propositionnel (sans variable). Soit l'ensemble d'atomes $E = \{val(x_1,\texttt{false})\}$. En utilisant le programme réduit $\Pi^1_{|E}$ et la définition par point fixe, montrez que E n'est pas un modèle stable de Π^1.

Question 3 Soit Q une 2QBF, nous disposons d'une fonction `varexist`(Q) qui retourne l'ensemble des variables quantifiées existentiellement dans Q, d'une fonction `renprogram`() qui crée un nouveau programme (vide) de REN et d'une fonction `addrule`(P, R) qui ajoute une règle, dont la représentation par string est R, à un programme P. Le traducteur qui génère à partir d'une 2QBF Q le programme (en REN) $\Pi^1(Q)$ dont les modèles stables contiennent toutes les valuations possibles des variables existentielles de Q (voir question 1) devrait ressembler à :

```
myprog = renprogram()
rule1 =
rule2 =
rule3 =
Pour nomvar in varexist(Q):
    addrule(myprog, rule1.format(nomvar))
    addrule(myprog, rule2.format(nomvar))
    addrule(myprog, rule3.format(nomvar))
    % j'utilise la méthode format de python
    % qui remplace {} dans une chaîne par son argument
    % par exemple "hello {}".format("world")
    % retourne "hello world"
return myprog
```

Écrire les 3 "patterns de règle" (`rule1`, ...) nécessaires pour compléter le traducteur. Ceci devrait beaucoup ressembler à votre réponse à la question 1.

Question 4 Soit la 2QBF $Q = \exists x \forall y (x \wedge y)$. Représentez l'arbre ASPERIX obtenu en faisant tourner le programme REN $\Pi^1(Q)$ (que vous expliciterez) généré par le traducteur de la question 4. Quels sont les modèles stables de ce programme? Vous justifierez soigneusement votre réponse.

Question 5 Si vous avez bien respecté les consignes de la question 1, votre programme $\Pi^1(Q)$ de la question 4 comporte une règle exprimant qu'une variable ne peut pas être valuée à la fois à `true` et à `false`. Cette règle est-elle utilisée dans la construction de votre arbre ASPERIX ? On pourrait penser que cette règle est inutile, et utiliser un traducteur ne générant pas cette règle dans le programme $\Pi^1_{v2}(Q)$. Montrez qu'on peut compléter ce programme avec des règles (ou des faits) de façon à obtenir un programme dont un modèle stable contient à la fois une valuation de x à `true` et une valuation de x à `false`. Vous justifierez soigneusement que ce programme répond bien à la question.

Question 6 Soit Q une 2QBF, et Π un programme quelconque contenant $\Pi^1_{v2}(Q)$ (voir question 5). Donnez une condition suffisante sur le programme Π permettant d'assurer que, pour chaque modèle stable de Π, il n'existe aucune variable existentielle de Q valuée à la fois à `true` et `false`. Vous pourrez vérifier à la fin du devoir que le programme final respecte cette condition, et donc que la règle interdisant la double valuation d'une variable était inutile.

4.3 Invalidité de la formule propositionnelle

Nous supposons maintenant que nous avons généré (via le traducteur Π^1) toutes les valuations possibles des variables existentielles $\vec{x}$ de $Q = \exists \vec{x} \forall \vec{y} Q'$, et nous souhaitons, dans le sous-arbre ASPERIX correspondant à une de ces valuations σ, générer $\bot$ (absurde) si la formule $\sigma(Q')$ est invalide. Ainsi, il ne restera que les modèles stables correspondant à une valuation σ pour laquelle $\sigma(Q')$ est valide, et nous aurons respecté la propriété demandée à notre transformation.

4.3.1 Analyse d'une conjonction

Pour tester l'invalidité de $\sigma(Q')$ (avec $Q' = c_1 \vee \ldots \vee c_k$), nous allons d'abord générer, pour chaque conjonction c_i, toutes les valuations σ_i qui étendent σ et pour lesquelles $\sigma_i(c_i) =$ `false`. Le calcul (par le traducteur) de ces valuations sera utilisé pour générer la partie $\Pi^2(Q)$ de notre programme.

Question 7 Soit la 3-conjonction $c_1 = x \wedge \neg y \wedge z$ apparaissant dans une 2QBF Q où x est quantifiée existentiellement (et y et z sont quantifiées universellement). Nous souhaitons écrire toutes les règles de la forme:

$$\texttt{fail/2}(c_1, y, V_y, z, V_z) : -val(x, V_x).$$

où les V sont instanciés par des booléens `true` où `false`, et où la règle signifie "si la variable existentielle x est valuée à V_x, alors valuer y par V_y et z par V_z entraîne une évaluation de la conjonction c_1 à `false`. Écrire les 7 règles (positives) générées par le traducteur à partir de Q permettant d'exprimer les cas d'échec de c_1.

Question 8 Même question avec la 3-conjonction c_1 de la question 7, mais apparaissant dans une 2QBF qui quantifie existentiellement x et y. Puisque nous n'avons besoin que de valuer une seule variable quantifiée universellement, nous aurons besoin d'un prédicat `fail/1`. Écrire les 5 règles (positives) permettant d'exprimer les cas d'échec de c_1.

Question 9 En supposant que toutes les conjonctions contiennent exactement 3 variables distinctes, comptez le nombre de règles nécessaires pour exprimer les règles générant respectivement `fail/0` (toutes les variables sont existentielles) et `fail/3` (toutes les variables sont universelles). Vous justifierez votre réponse en donnant la forme des règles obtenues.

4.3.2 C'est la contrainte finale

Pour des raisons de temps, je ne vous ai pas demandé d'écrire le traducteur générant les règles du programme $\Pi^2(Q)$ telles que vous les avez écrites aux questions 7-9.

Question 10 Reprenons l'exemple F de la section 4.1. Écrire le programme $\Pi^1(F) \cup \Pi^2(F)$ associé à cette formule (vous pourrez ne donner que l'échantillon des règles de $\Pi^2(F)$ nécessaire pour répondre à la question 12).

Question 11 Quelle contrainte négative (qui serait générée automatiquement par le traducteur $\Pi^3(Q)$) faut-il ajouter pour tester si $\sigma(F')$ est insatisfiable dans la branche correspondant à la valuation σ ?

Question 12 Montrez que la branche de l'arbre ASPERIX correspondant à la valuation $\sigma = \{x : \texttt{true}, y : \texttt{true}\}$ (vous pourrez en faire la racine de l'arbre) ne mène pas à un modèle stable.