Université de Montpellier - Master informatique	Janvier 2022
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 réponse sans justification ne sera pas prise en compte. Le barème est donné à titre indicatif et peut varier légèrement.

Exercice 1 (Homomorphismes) - 1 pt

Soient deux ensembles d'atomes, où a est la seule constante :
$Q_1 = \{p(a, x_1), q(x_1, y_1)\}$
$Q_2 = \{p(x_2, y_2), q(y_2, z_2), q(z_2, u_2)\}$

Question 1 Existe-t-il un homomorphisme de Q_1 dans Q_2? De Q_2 dans Q_1?

Question 2 Si Q_1 et Q_2 représentent des requêtes conjonctives booléennes, a-t-on $Q_1 \sqsubseteq Q_2$? $Q_2 \sqsubseteq Q_1$?

Rappel: Etant données deux requêtes conjonctives booléennes Q et Q', la notation $Q \sqsubseteq Q'$ signifie que toute base de faits qui répond oui à Q répond aussi oui à Q'.

Exercice 2 (Chase) - 4,5 pts

Soit la base de connaissances $\mathcal{K} = (F, \mathcal{R} = \{R_1, R_2\})$ avec :
$F = \{p(a, b), r(a)\}$
$R_1 = p(x, y) \to \exists z\ p(y, z)$
$R_2 = r(x) \wedge p(x, y) \to p(y, y)$.

Nous allons considérer trois variantes du chase :

- l'*oblivious* chase, qui effectue toutes les applications de règles possibles ;
- le *restricted* chase, qui n'effectue l'application d'une règle $R = B \to H$ selon un homomorphisme h de B dans la base de faits courante F que si h ne *s'étend pas* à un homomorphisme de $B \cup H$ dans F ;
- le *core* chase, qui applique les règles comme le restricted chase, mais calcule le *core* de la base de faits obtenue après chaque application de règle.

Question 1 L'*oblivious chase* s'arrête-t-il sur cette base de connaissances ? Pourquoi ? (On ne vous demande pas de donner le résultat du chase).

Question 2 Définir le résultat obtenu par le *restricted chase.* Si l'ordre d'application des règles change le résultat, considérez les différents cas possibles.

Question 3 Définir le résultat obtenu par le *core chase.*

Question 4 La base de connaissances $\mathcal{K}$ possède-t-elle un modèle universel fini ?

Question 5 L'ensemble de règles $\mathcal{R}$ assure-t-il que *pour toute base de faits* F, $(F, \mathcal{R})$ a un modèle universel fini?

Exercice 3 (First-order Queries) - 2 pts

Considérez la base de données suivante :

Films

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

Cinémas

Cinéma	Addresse	Téléphone
UFA	St. Petersburger Str. 24	4825825
Schauburg	Königsbrücker 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” (un demi-point chacune) :

1. Qui sont les réalisateurs qui ont travaillé avec l'acteur “Cumberbatch” ?
2. Lister les réalisateurs qui ont réalisé un film projeté au cinéma “UFA”.
3. Découvrez tous les acteurs qui n'ont pas “Tyldum” comme réalisateur.
4. Ecrivez une requête booléenne (Boolean query) pour déterminer si deux réalisateurs différents ont réalisé le même film.

Exercice 4 (Join Trees) - 2 pts

Considérez les requêtes suivantes :

1. $\exists x, y, z, w.P(x, y, z) \wedge Q(x, z, w) \wedge R(x, x, y, w) \wedge P(y, w, z)$
2. $\exists z_1, \ldots, z_6.V(z_1, z_2, z_6) \wedge O(z_1, z_2, z_3) \wedge M(z_2, z_4, z_3) \wedge L(z_1, z_3, z_5)$

Si possible, définissez un “join tree” pour chacune de ces requêtes. Si ce n'est pas possible de le faire, expliquez pourquoi.

Exercice 5 (OWA/CWA) - 2,5 pts

On considère des bases de faits et requêtes conjonctives munies de la négation. On se place dans un premier temps en monde ouvert, autrement dit dans le cadre de la logique classique.

Soit la base de faits $F = \{q(a), q(b), p(a, a), \neg p(b, b)\}$ où a et b sont des constantes.

Question 1. Soit $Q_1() = \exists x \exists y \exists z\ (p(x, y) \wedge q(z) \wedge \neg p(y, z))$. Déterminer $Q_1(F)$, l'ensemble des réponses à Q_1 sur F.

Question 2. Même question avec $Q_2(x) = \exists y \exists z\ (p(x, y) \wedge q(z) \wedge \neg p(y, z))$.

Question 3. Même question avec $Q_3(y) = \exists x \exists z\ (p(x, y) \wedge q(z) \wedge \neg p(y, z))$.

Question 4. On considère la base de faits $F' = \{q(a), q(b), p(a, a)\}$ et on fait l'hypothèse du monde clos. Quelles sont alors les réponses à Q_1, Q_2 et Q_3 sur F' ?

Exercice 6 – Modèles stables – 4 pts

Question 1 Utilisez la méthode vue en cours pour mettre le programme suivant sous forme propositionnelle. Vous détaillerez soigneusement les étapes de cette transformation.

$$p(a), q(a, b).$$
$$p(X), \text{not } q(X, Y) \rightarrow r(X).$$

Détaillez le calcul d'une dérivation persistante et complète utilisant les règles que vous avez obtenues.

Remarque : si le résultat de cette dérivation n'est pas conforme à votre intuition, peut-être avez vous oublié une étape lors de votre transformation.

Question 2 Nous considérons maintenant le programme Π défini ci-dessous (7 lignes au total), et souhaitons savoir si les ensembles d'atomes $E_1 = \{a, c, d, f\}$ et $E_2 = \{a, e, f\}$ sont des modèles stables de ce programme.

$$
\begin{array}{ll}
a. & \\
a, \text{not}\, c \rightarrow b. & b \rightarrow c. \\
a \rightarrow f. & f, \text{not}\, e \rightarrow d. \\
f, \text{not}\, d \rightarrow e. & d \rightarrow c.
\end{array}
$$

Vous construirez les programmes réduits associés à ces deux ensembles d'atomes et les utiliserez pour répondre à la question en utilisant la définition par point fixe.

Question 3 Reprenons le programme Π de la question 2. Les ensembles d'atomes $E_4 = \{a, c, d, e, f\}$, et $E_5 = \{c, d, e, f\}$ sont-ils des modèles stables de ce programme? Votre réponse devra être argumentée, mais n'utilisera pas nécessairement la définition par point fixe.

Question 4 Utilisez l'algorithme ASPERIX vu en cours pour trouver tous les modèles stables du programme Π de la question 2. *Vous vérifierez que vos réponses aux questions précédentes sont cohérentes.* Considérant le nombre de modèles stables que vous avez trouvé, pouvez-vous en déduire si le programme Π est stratifiable ?

Exercice 7 – Modélisation en ASP – 4 pts

Les points sont accordés si au moins la moitié de l'exercice est correctement traitée.

Dans le jeu "Le compte est bon", il faut réussir à obtenir un nombre cible à partir de 6 nombres initiaux, en utilisant les 4 opérations arithmétiques élémentaires: addition, soustraction, multiplication, division. Les conditions initiales du jeu seront codées comme dans l'exemple suivant :

$$
\begin{array}{l}
\text{disponible}(a, 25, 0). \\
\text{disponible}(b, 10, 0). \\
\text{disponible}(c, 10, 0). \\
\text{disponible}(d, 7, 0). \\
\text{disponible}(e, 3, 0). \\
\text{disponible}(f, 4, 0). \\
\text{cible}(284).
\end{array}
$$

L'atome $\text{disponible}(a, 25, 0)$. se lit *"Le nombre qui a pour identifiant a, dont la valeur est 25, est disponible à l'étape 0"*. Coder uniquement le fait que *"le nombre 25 est disponible à l'étape 0"* ne permettrait pas d'avoir des occurences multiples d'un même nombre. En supposant qu'on ait pu choisir 2 nombres et une opération à chaque étape, 4 règles conçues sur le modèle de la règle suivante (qui concerne l'addition) permettent de générer un nouveau nombre à l'étape N+1 :

$$
\begin{array}{l}
\text{choisi1}(X1, N), \text{choisi2}(X2, N), \text{opchoisie}(\text{addition}, N), \\
\text{disponible}(X1, V1, N), \text{disponible}(X2, V2, N) \rightarrow \text{disponible}(Y, V1 + V2, N + 1).
\end{array}
$$

Le but de l'exercice est d'écrire un programme dont les modèles stables encodent toutes les manières d'arriver au nombre cible. Il s' agit d'une version simplifiée du jeu, en particulier, on ne cherche pas à générer un nombre "le plus proche possible" de la cible.

Question 1 Ecrire une règle permettant de générer l'atome $fini()$ dès que la valeur cible a été trouvée. Complétez ensuite le programme pour que les modèles stables contiennent uniquement les cas où cette valeur cible a été trouvée.

Question 2 L'atome $choisi1(X, N)$ signifie que le nombre d'identifiant X a été choisi comme premier argument de notre calcul à l'étape N. Ecrire la (ou les) règle(s) permettant de générer cet atome. Vous vous inspirerez du choix multiple vu en cours, et prendrez en compte les points suivants :

- il n'y a plus besoin de faire de choix quand le jeu est fini ;
- on ne peut pas faire de choix quand on n'a pas au moins deux identifiants de nombres disponibles ;
- un seul identifiant de nombre peut être choisi.

Question 3 L'atome $choisi2(X, N)$ signifie que le nombre d'identifiant X a été choisi comme deuxième argument de notre calcul à l'étape N. Ecrire la (ou les) règle(s) permettant de générer cet atome. Vous vous inspirerez du choix multiple vu en cours, et prendrez en compte les points suivants:

- on ne fait le choix 2 qu'après le choix 1;
- on ne peut pas faire le même choix en choix2 qu'en choix1 ;
- un seul identifiant de nombre peut être choisi.

Question 4 L'atome $opchoisie(X, N)$ signifie que l'opération X a été choisie à l'étape N. Ecrire la (ou les) règle(s) permettant de générer cet atome. Vous vous inspirerez du choix multiple vu en cours, et prendrez en compte les points suivants :

- on ne peut choisir que parmi les opérations codées dans les atomes suivants : $op(addition), op(soustraction), op(multiplication), op(division)$;
- il n'y a plus besoin de faire de choix quand le jeu est fini ;
- une seule opération peut être choisie à chaque étape ;
- (OPTIONNEL) on ne peut choisir la division que si son résultat est un nombre entier : on pourra tester dans le corps de la règle $X1\%X2 == 0$ pour savoir si la valeur $X1$ du premier nombre choisi est divisible par la valeur $X2$ du second.

Question 5 Ecrire la ou les règles permettant d'assurer que les nombres qui n'ont pas été choisis à une étape sont encore disponibles à l'étape suivante.

Question 6 L'algorithme ASPERIX s'arrête-t-il sur votre programme? Justifiez votre réponse. Montrez que le grounding de ce programme est infini. Identifiez-en toutes les causes. Proposez une modification de votre programme pour obtenir un grounding fini.