UNIVERSITÉ DE MONTPELLIER
-
FACULTÉ DES SCIENCES

Session : 1 Durée de l'épreuve : 3 heures

Date : Janvier 2020

Documents autorisés : oui

Master Informatique

Théorie des bases de connaissances (HMIN312)

Barème indicatif (/ 21) : Ex. 1 : 6 ; Ex. 2 : 5 (+) ; Ex. 3 : 4 ; Ex 4 : 3 ; Ex. 5 : 3 (+)

Merci de rédiger les exercices 1+2 et 3+4+5 sur des copies séparées.

Exercice 1. Règles existentielles

On considère la base de connaissances $\mathcal{K} = (F, \mathcal{R})$ avec $F = \{r(a)\}$, où a est une constante, et $\mathcal{R} = \{R_1, R_2\}$, où :
$R_1 : r(x) \rightarrow \exists z\ p(x, z)$
$R_2 : p(x, y) \rightarrow \exists z\ p(x, z) \wedge q(z)$

Question 1.1 Définir la base de faits saturée (éventuellement infinie) obtenue à partir de F et $\mathcal{R}$ selon les quatre types de *chase* suivants, qu'on supposera être exécutés en largeur :

1. oblivious ;
2. semi-oblivious ;
3. restricted ;
4. core.

Question 1.2 On considère l'interprétation I dont le domaine est $D = \{a, z\}$ (avec la simplification adoptée en cours : chaque constante est interprétée par un élément du domaine de même nom) et telle que :
$I(r) = \{a\}$ $I(q) = \{z\}$ $I(p) = \{(a, z)\}$

(a) Montrer que I est un modèle de $\mathcal{K} = (F, \mathcal{R})$.
(b) Est-ce un modèle universel de $\mathcal{K}$? Justifier.
(c) Une base de connaissances peut-elle avoir plusieurs modèles universels ? En particulier, combien la base $\mathcal{K}$ de cet exercice a-t-elle de modèles universels ? Justifier votre réponse.

Question 1.3 L'ensemble de règles $\mathcal{R}$ assure-t-il que pour *toute* base de faits F', la base de connaissances $(F', \mathcal{R})$ a un modèle universel fini ? Autrement dit, $\mathcal{R}$ est-il un "finite expansion set " ? Justifier votre réponse.

Question 1.4
(a) Donner un modèle de $\mathcal{K}$ ayant le plus petit domaine possible.
(b) Montrer que ce modèle n'est pas un modèle universel de $\mathcal{K}$.

Question 1.5
(a) On rappelle qu'un modèle universel joue le rôle de "représentant" de tous les modèles d'une base de connaissances lorsqu'on cherche à répondre à des requêtes conjonctives. En effet, étant donnée une requête conjonctive booléenne q, on a : $\mathcal{K} \models q$ si et seulement si l'un quelconque des modèles universels de $\mathcal{K}$ est un modèle de q. Donner une requête conjonctive qui montre que le modèle que vous avez construit en 4.(a) n'est pas universel.
(b) La propriété "$\mathcal{K} \models q$ si et seulement si l'un quelconque des modèles universels de $\mathcal{K}$ est un modèle de q" est-elle conservée si q est une requête booléenne *avec négation du monde clos* ? Ilustrer votre réponse sur la base de connaissances $\mathcal{K}$ de l'exercice.

Exercice 2. Réécriture de requêtes conjonctives

Soit la base de règles $\mathcal{R} = \{R_1, R_2, R_3\}$ avec :
$R_1 = q(x, y), s(y) \rightarrow \exists z\ p(y, z)$
$R_2 = p(x, y) \rightarrow q(x, y)$
$R_3 = p(x, y) \rightarrow \exists z\ r(y, z)$

Soit la requête conjonctive booléenne $q = p(u, v), p(v, w), r(w, t)$.

Question 2.1 Quel est l'ensemble $\mathcal{Q}$ de toutes les réécritures *non isomorphes* de q avec $\mathcal{R}$? Dessiner l'arbre des réécritures de q en indiquant la règle et l'unificateur utilisé pour produire chaque réécriture, et en s'arrêtant sur les réécritures qui sont isomorphes à une réécriture déjà obtenue. Lorsque vous obtenez une requête qui est intrinsèquement redondante (c'est-à-dire qui n'est pas un core), vous la remplacerez par son core avant de continuer.

Question 2.2 Existe-t-il un sous-ensemble strict de $\mathcal{Q}$ qui reste adéquat et complet ? Si oui, donnez un tel ensemble qui soit minimal au sens de l'inclusion.

Question 2.3 On suppose maintenant que les bases de connaissances considérées comportent aussi des contraintes négatives, et que l'on n'interroge que des bases de connaissances satisfiables (consistantes).
(a) Considérons l'ensemble $\mathcal{R}$ ci-dessus et ajoutons la contrainte négative suivante :

$$C : q(x, y), s(y) \rightarrow \bot$$

Lorsqu'on réécrit la requête q ci-dessus, on sait que toute base de connaissances $(F, \mathcal{R}, \{C\})$ interrogée avec q est consistante. Peut-on exploiter cette connaissance dans le calcul des réécritures de q ? Quel serait alors l'ensemble $\mathcal{Q}$ obtenu ?
(b) De façon générale, comment exploiter dans le calcul des réécritures d'une requête la présence de contraintes négatives et le fait que toute base de connaissances interrogée est satisfiable ?

Exercice 3 : Modèles stables et triple choix

On considère le programme propositionnel Π suivant :

(R1) h.
(R2) h, not c2, not c3 → c1.
(R3) h, not c1, not c3 → c2.
(R4) h, not c1, not c2 → c3.

Dans les questions suivantes, vous pourrez utiliser la règle :

(R5) abs, not abs_aux → abs_aux.

Ainsi, si un programme Π′ contient la règle $R5$ et aucune règle de Π′ ne peut générer abs_aux, alors aucun modèle stable de $R5$ ne peut contenir abs, qui simule l'atome absurde.

Question 3.1 A l'aide de l'algorithme ASPERIX vu en cours (arbre de dérivations), énumérez tous les modèles stables du programme Π.

Question 3.2 Montrez que l'affirmation suivante est fausse :

Affirmation 1. *Soit un programme* Π′ *contenant le programme* Π. *Alors chacun des atomes c1, c2 et c3 appartient à un modèle stable (ces modèles pouvant être différents ou non).*

Pour cette démonstration, vous donnerez le programme Π′ (vous pouvez ajouter à Π des faits ou des règles), et prouverez à l'aide de la méthode de votre choix qu'il ne satisfait pas l'affirmation.

Question 3.3 Montrez que l'affirmation suivante est fausse :

Affirmation 2. *Soit un programme* Π′ *contenant le programme* Π. *Alors tout modèle stable de* Π′ *contient au plus l'un des atomes c1, c2 ou c3.*

Pour cette démonstration, vous donnerez le programme Π′, un ensemble d'atomes ne respectant pas les conditions de l'affirmation, et prouverez qu'il s'agit d'un modèle stable en utilisant la définition par point fixe (basée sur le programme réduit).

Question 3.4 On souhaite maintenant modifier le programme Π de façon à respecter l'affirmation 2, afin de simuler un *ou exclusif*. Quelle(s) règle(s) faut-il rajouter à notre programme ?

Question 3.5 Le programme Π est-il stratifiable ? Soit Π′ un programme contenant Π admettant un unique modèle stable. Pouvez-vous affirmer que Π′ est stratifiable ou qu'il ne l'est pas ? Vous justifierez soigneusement votre réponse.

Exercice 4 : Modèles stables et choix multiples

Nous avons vu en cours un programme "si h alors c1 ou c2", et dans l'exercice précédent un programme "si h alors c1 ou c2 ou c3". Ce programme pourrait facilement se généraliser à "si h alors c1 ou c2 ou ... ou cn". Cependant on souhaite écrire des ensembles de règles indépendants des faits. Par exemple, si on veut choisir un sommet dans un graphe, on veut que l'ensemble de règles fonctionne indépendamment du nombre de sommets du graphe.

Dans tout le reste de ce devoir, nous nous plaçons dans le cadre de la logique du premier ordre avec hypothèse du nom unique. En particulier, vous aurez droit d'utiliser le prédicat $\neq$, et l'atome $a \neq b$ sera vrai dès que a et b sont deux constantes distinctes. Comme convention, les noms de variable commenceront par une majuscule et les constantes par une minuscule.

Question 4.1 Dans une première version de notre programme, imaginons un programme qui aurait le comportement suivant :

- le predicat unaire choix_poss (pour possible) indique les constantes parmi lesquelles nous avons à choisir (par exemple nous avons dans la base de faits choix_poss(c1), choix_poss(c2), choix_poss(c3)) ;
- on veut que si l'atome faire_choix() (d'arité 0) appartient à un modèle stable, alors, en l'absence d'autres règles (voir exercice 1), ce modèle stable contient l'atome *choisi(a)*, où a est une des constantes indiquées comme choix possible (dans notre exemple, on aura soit choisi(c1), soit choisi(c2), soit choisi(c3) dans notre modèle stable).

A cet effet, nous commençons par écrire le programme suivant :

(S1) faire_choix(), choix_poss(X), not choisi(X) $\rightarrow$ choisi(X)

Expliquez pourquoi ce programme n'a pas l'effet souhaité (vous pourrez, par exemple, faire tourner l'algorithme ASPERIX avec un seul choix possible).

Question 4.2 Pour répondre à ce problème, nous essayons le programme suivant :

(T1) faire_choix(), choix_poss(X), choix_poss(Y), X $\neq Y$, not choisi(Y) $\rightarrow$ choisi(X)

Expliquez pourquoi ce programme n'a pas l'effet souhaité (vous pourrez, par exemple, faire tourner l'algorithme ASPERIX avec trois choix possibles en ne donnant que la branche qui illustre le problème).

Question 4.3 Complétez le programme suivant de façon à ce qu'il ait le comportement voulu (ici, la sémantique intuitive de autre_choix(X) est qu'un autre choix que X a été effectué) :

(U1) faire_choix(), choix_poss(X), not autre_choix(X) $\rightarrow$ choisi(X)
(U2) <*mettre ici le bon corps de règle*> $\rightarrow$ autre_choix(X)

Question 4.4 Faites tourner ASPERIX sur votre programme de la question précédente (avec 3 choix possibles : c1, c2 et c3). Assurez vous qu'il a bien le comportement voulu. L'arbre n'aura pas besoin d'être complet mais il faudra montrer, par exemple sur c1, qu'il y a un modèle stable qui contient choisi(c1) et pas choisi(c2) ni choisi(c3). Vous démontrerez alors que tout modèle stable contenant choisi(c1) ne peut contenir ni choisi(c2) ni choisi(c3).

Question 4.5 Un problème de notre programme est qu'il n'est capable d'effectuer qu'un seul choix pendant toute l'exécution du programme, ce qui est très limitatif. Une solution peut-être de rajouter un argument à chaque prédicat (cet argument peut être vu comme l'étape à laquelle on effectue ce choix). Le programme devient alors :

(U1) faire_choix(N), choix_poss(X, N), not autre_choix(X, N) $\rightarrow$ choisi(X, N)
(U2) <*mettre ici le bon corps de règle*> $\rightarrow$ autre_choix(X, N)

Complétez le progamme et rajoutez une (ou plusieurs) règle(s) permettant de simuler qu'au plus un choix est possible à chaque étape (vous pourrez vous inspirer de la question 3.4).

Exercice 5 : Modèles stables et circuits Hamiltoniens

Un graphe non orienté peut être représenté dans une base de faits par les atomes suivants (ici un graphe complet à 3 sommets) :

(Ex1) sommet(a), sommet(b), sommet(c), arete(a, b), arete(a, c), arete(b, c).

Un *circuit hamiltonien* est un circuit du graphe qui visite chaque sommet une fois et une seule avant de revenir à son point de départ. Dans cet exercice, nous allons utiliser le programme de l'exercice 4 pour écrire un programme dont chaque modèle stable encode un circuit hamiltonien. Par exemple, si a, b, c, a est un circuit hamiltonien, le modèle stable contiendra les atomes choisi(a, 0), choisi(b, 1), choisi(c, 2), choisi(a, 3).

Question 5.1 Ecrire une règle encodant la symétrie des arêtes dans un graphe non orienté.

Question 5.2 Comme la recherche d'un circuit hamiltonien ne dépend pas du sommet par lequel on commence, on peut supposer que l'utilisateur choisit le sommet de départ en rajoutant dans la base de faits (par exemple si on veut partir du sommet a) :

(Ex2) choisi(a, 0).
(Ex3) faire_choix(1)

A chaque autre étape, il faudra choisir le sommet suivant parmi tous les sommets voisins de celui choisi à l'étape précédente. Ecrire la ou les règles permettant d'effectuer ce choix. Nous devrons exclure des choix possibles les sommets déjà choisis dans le circuit. Pour cela, nous supposons pour l'instant que nous disposons du prédicat deja_choisi : l'atome deja_choisi(X, N) veut dire que le choix de X a déja été fait avant l'étape N.

Question 5.3 Ecrire une ou plusieurs règle(s) permettant de générer les atomes déja_choisi(X, N) nécessaires.

Question 5.4 L'atome fini(N) indique que tous les sommets ont été choisis à l'étape N, et donc que tous les choix ont été correctement faits. Ecrire une ou plusieurs règle(s) permettant de générer cet atome.

Question 5.5 Enoncez en français des connaissances qui manquent encore dans cette modélisation pour générer tous les circuits hamiltoniens d'un graphe, et donnez les règles correspondantes. En particulier, vous penserez à incrémenter l'étape de choix après avoir choisi un sommet et vous devrez gérer les particularités du dernier sommet du circuit (il n'y a plus de choix à faire, et il a déja été choisi au début).