Université de Montpellier - Master informatique Janvier 2021

Théorie des bases de connaissances (HMIN 312)

Examen

Durée totale : 2 heures
Répartition conseillée : 40 à 45 mn pour la partie 1

PARTIE 1 : REGLES EXISTENTIELLES (notation / 20)

Exercice 1 (Chase) - 13 pts

Soit la base de connaissances $\mathcal{K}_1 = (F_1, \mathcal{R}_1 = \{R_1, R_2\})$ avec :
$F_1 = \{p(a,b), r(a)\}$
$R_1 = p(x,y) \rightarrow \exists z\ p(y,z)$
$R_2 = r(x) \wedge p(x,y) \rightarrow p(y,y)$

Question 1 Définir la saturation de F_1 par $\mathcal{R}_1$ en utilisant l'*oblivious chase*.

Question 2 Définir la saturation de F_1 par $\mathcal{R}_1$ en utilisant le *restricted chase*. Si l'ordre d'application des règles change le résultat, considérez les différents cas possibles.

Question 3 Définir la saturation de F_1 par $\mathcal{R}_1$ en utilisant le *core chase* (on supposera que la base de faits courante est mise sous la forme d'un core à chaque application de règle).

Question 4 La base de connaissances $\mathcal{K}_1$ possède-t-elle un modèle universel fini ? Justifier votre réponse.

Question 5 L'ensemble de règles $\mathcal{R}_1$ est-il un "finite expansion set", c'est-à-dire assure-t-il que pour toute base de faits F, $(F, \mathcal{R}_1)$ a un modèle universel fini?

Question 6 Utilisez la base de connaissances $\mathcal{K}_2$ ci-dessous pour montrer que le core chase est *strictement* plus puissant que le restricted chase en termes de terminaison.

$\mathcal{K}_2 = (F_2, \mathcal{R}_2 = \{R_1, R_3\})$ avec :
$F_2 = \{p(a,b), r(b)\}$
$R_1 = p(x,y) \rightarrow \exists z\ p(y,z)$
$R_3 = r(x) \wedge p(x,y) \rightarrow p(x,x)$

Exercice 2 (Règles existentielles linéaires) - 7 pts

Soit l'ensemble de règles $\mathcal{R} = \{R_1, R_2\})$ avec :
$R_1 = p(x,y) \rightarrow \exists z\ p(y,z) \wedge q(z)$
$R_2 = s(x,y) \rightarrow \exists z\ p(y,z) \wedge t(z)$.

Question 1 Soit la requête conjonctive booléenne $q() = \exists u \exists v\ p(u,v) \wedge t(u) \wedge t(v)$. Donner l'ensemble des réécritures de q avec $\mathcal{R}$. Expliquer pourquoi il n'y a pas d'autres réécritures.

Question 2 Une règle existentielle est dite *linéaire* si son corps est composé d'un seul atome (comme R_1 et R_2). Montrer que l'ensemble des réécritures (non-isomorphes) d'une requête conjonctive booléenne avec un ensemble de règles linéaires est toujours fini. On rappelle que deux ensembles d'atomes sont isomorphes si on passe de l'un à l'autre par un renommage bijectif de leurs variables (q_1 et q_2 sont isomorphes s'il existe une bijection b de l'ensemble des variables de q_1 dans l'ensemble des variables de q_2 telle que $b(q_1) = q_2$).

PARTIE 2 : MODELISATION EN ASP (notation / 20)

Le but de cette modélisation est de coder en ASP la possibilité pour un agent de mentir ou de dire la vérité, et de raisonner sur les affirmations de ces agents. Les exemples d'application de cette modélisation sont tirés du livre de Raymond Smullyan: Le livre qui rend fou.

Exercice 1 (Méta-modélisation: les bases) – 5 pts

Question 1 Nous utiliserons pour l'instant les prédicats suivants:

`say(A, C)`: l'agent A énonce l'assertion (claim) C

`truth(C)`: l'assertion C est véridique (truthful)

`lie(C)`: l'assertion C est un mensonge.

Exprimez par un programme ASP Π_{base} le fait que si un agent dit quelque chose ("énonce une assertion"), alors soit cette chose est véridique, soit c'est un mensonge. Bien entendu, une assertion (claim) ne peut pas être à la fois véridique et un mensonge.

Question 2 Ce programme Π_{base} est-il stratifiable? Justifiez votre réponse.

Question 3 Le programme Π_1 est obtenu à partir du programme Π_{base} en rajoutant les atomes `say(bob, c), lie(c)`. Par la méthode vue en cours, transformez le programme Π_1 en un programme propositionnel Π_1'. Vous justifierez chacune des étapes de cette transformation. **Lors du passage en propositionel, vous utiliserez la convention suivante: un atome `pred(orig, dest)` devient un atome propositionnel `pod` en concaténant les initiales.**

Si vous n'avez pas trouvé le programme Π_1', vous pourrez dans les questions qui suivent le remplacer par le programme suivant (qui n'est pas tout à fait Π_1').

```
sbc, lc.

tc :- sbc, not lc.

lc :- sbc, not tc.

abs :- lc, tc, not abs.
```

Question 4 En utilisant la méthode du point fixe, dire si {`sbc, lc`} est un modèle stable de Π_1'.

Question 5 En justifiant de la manière la plus rapide possible (c'est à dire en évitant la méthode du point fixe), dire si {`sbc, tc`}, {`sbc`} et {`sbc, tc, lc`} sont des modèles stables de Π_1'.

Question 6 Combien Π_1' a-t-il de modèles stables? Pouvez-vous vous servir de la réponse à cette question pour répondre à la question 2?

Exercice 2 (Méta-modélisation: liaison avec le réel) – 5 pts

Nous voulons maintenant lier les assertions des agents avec ce qui est vrai ou faux dans le monde considéré. Pour l'instant, une assertion sera uniquement un triplet `(s, p, o)` (sujet, prédicat, objet, à la façon RDF). *Pour ceux qui ne connaissent pas RDF, un triplet* `(s, p, o)` *encode un atome de prédicat binaire* `p(s, o)`*. Par exemple, le triplet* `(princess, in, cell1)` *est une notation alternative pour l'atome* `in(princess, cell1)` *et indique qu'une princesse se trouve dans la cellule 1 (voir plus tard pour l'application).* Nous introduisons maintenant les prédicats suivants:

`claim(C, S, P, O)`: l'assertion (claim) C exprime le triplet `(S, P, O)`

`triple(S, P, O)`: le triplet `(S, P, O)` est vrai dans ce monde.

Question 7 Pour lier les assertions au réel, il nous faudra coder les connaissances suivantes:

- si une assertion exprime un triplet qui est vrai, alors cette assertion est véridique ;
- si une assertion exprime un triplet qui est faux (monde clos), alors cette assertion est un mensonge ;
- si une assertion véridique exprime un triplet, alors il est vrai ;
- si une assertion mensongère exprime un triplet, alors il est faux.

Les règles exprimant ces connaissances, ajoutées à celles du programme Π_{base}, forment le programme Π_{reel}.

Question 8 Dans l'application finale de notre modélisation, il sera question de deux cellules de prison qui contiennent chacune un tigre ou une princesse. Nous modélisons ceci par le programme Π_{prison} suivant (où le symbole != indique la différence):

```
cell(cell1;cell2).  %% macro pour cell(cell1).  cell(cell2).

entity(tiger;princess).

triple(E', in, C) :- cell(C), entity(E), entity(E'), E != E', not triple(E, in,
C).

:- triple(E, in, C), triple(E', in, C), E != E'. %% on rappelle qu'en ASP, les
atomes en tête sont traités comme une disjonction, et que la disjonction vide
veut dire absurde.
```

Déroulez l'algorithme ASPERIX (au brouillon) sur ce programme Π_{prison} pour répondre aux questions suivantes: peut-on avoir un tigre et une princesse dans la même cellule? Une cellule peut-elle être vide?

Question 9 On considère maintenant le programme Π_2 composé des programmes Π_{reel} et Π_{prison}, ainsi que des atomes: `say(affiche1, a1).  say(affiche2, a2).  claim(a1, princess, in, cell1).  claim(a2, tiger, in, cell2).  lie(a1).  truth(a2)`. Déroulez l'algorithme ASPERIX sur ce programme Π_2. Attention, certains arbres peuvent être beaucoup plus grands que d'autres!

Exercice 3 (Méta-modélisation: connecteurs logiques) – 5 pts

Pour l'instant, la seule assertion qu'un agent peut faire est un triplet, ce qui ne sera pas suffisant pour résoudre les problèmes amusants du livre de Raymond Smullyan. Nous nous proposons d'y ajouter quelques connecteurs logiques.

Question 10 Soit le programme Π_{conj} suivant (où `conjunction(C, C1, C2)` veut dire que `C` est la conjonction de `C1` et de `C2`), qui exprime la sémantique de la conjonction d'assertions:

- truth(C) :- conjunction(C, C1, C2), truth(C1), truth(C2).
- truth(C1) :- conjunction(C, C1, C2), truth(C).
- truth(C2) :- conjunction(C, C1, C2), truth(C).
- lie(C2) :- conjunction(C, C1, C2), lie(C), not lie(C1).
- lie(C1) :- conjunction(C, C1, C2), lie(C), not lie(C2).

Faites tourner l'algorithme ASPERIX sur le programme Π_3 obtenu à partir des programmes Π_{reel} et Π_{prison}, auxquels on aura rajouté les atomes suivants: `say(affiche, a).` `conjunction(a, a1, a2).` `claim(a1, princess, in, cell1).` `claim(a2, tiger, in, cell2).` `lie(a).` Que se serait-il passé si on n'avait pas utilisé les règles de la question 8 (Π_{prison})?

Question 11 De façon similaire à la question 10, donnez le sous-programme Π_{disj} codant la disjonction d'assertions.

Question 12 De façon similaire à la question 10, donnez le sous-programme Π_{neg} codant la négation d'une assertion.

Exercice 4 (Application: la première épreuve) – 5 pts

Dans le chapitre 2 de son livre *Une princesse ou un tigre?*, Smullyan propose des petits problèmes logiques tous basés sur le même principe. Un prisonnier est en face de deux cellules, dont chacune contient une princesse ou un tigre. Sur chaque cellule est affichée une phrase logique, du type "cette cellule contient une princesse". Ces phrases peuvent être vraies ou fausses, et le Roi donne un indice final sur leur véracité (comme "au moins une des 2 affiches dit vrai"). Le prisonnier doit deviner où est la princesse. Au cours des questions précédentes, nous avons tout mis en place pour résoudre automatiquement une grande partie de ces problèmes grâce à ASP. Nous nous proposons ici de résoudre "la première épreuve". Nous utiliserons toutes les règles des programmes $\Pi_{reel}, \Pi_{prison}, \Pi_{conj}, \Pi_{disj}$ et Π_{neg}.

Question 13 Sur la première cellule `cell1` est affichée l'assertion suivante: *"Il y a une princesse dans* **cette** *cellule et un tigre dans l'autre."*. Complétez `say(affiche1, a1).` pour modéliser cette affirmation.

Question 14 Sur la deuxième cellule `cell2` est affichée l'assertion suivante: *"Il y a une princesse dans* **une** *cellule et un tigre dans l'autre."*. Complétez `say(affiche2, a2).` pour modéliser cette affirmation.

Question 15 Le roi (qui ne ment jamais, donc pas besoin de modéliser cette connaisssance par une assertion "say"), dit que *"l'une des 2 affiches dit la vérité, et l'autre ment"*. Modélisez cette connaissance.

Pour aller plus loin...

Tout d'abord félicitations! Si vous avez correctement tout modélisé, le solveur clingo va énumérer en 0.015s l'unique modèle stable de ce problème, qui contient les atomes `triple(tiger, in, cell1).` `triple(princess, in, cell2).` Le prisonnier peut donc choisir sereinement, ne sera pas mangé, et gagnera un mariage qu'on espère heureux.

Si vous êtes arrivé juqu'ici, et que vous avez encore un peu de temps, les questions suivantes sont des "questions bonus". Elles ne sont pas comptabilisées dans le barême, mais peuvent vous donner des points supplémentaires. Vous pouvez y répondre tout de suite, ou même m'envoyer vos réponses par mail (`baget@lirmm.fr`) avant la fin de la semaine.

Question 16 *(pas très dur, peut être fait pendant l'examen)* La conjonction (comme la disjonction) que nous avons écrite est une conjonction binaire. Bien entendu, nous pourrions écrire une conjonction n-aire par une succession de conjonctions binaires (faites-le pour une conjonction de 3 arguments). Mais nous préfererions écrire quelque chose du genre : `conjunction(a). argument(a, a1). argument(a, a2). argument(a, a3).` Ecrivez un programme qui gère la sémantique de la conjonction avec cette syntaxe.

Question 17 *(bien plus dur, ceux qui ont le temps pourront faire ce qu'ils peuvent pendant l'examen, mais je m'attends à recevoir des réponses par mail – vous pourrez joindre le fichier q17.lp que vous aurez testé sous clingo)* Nous utiliserons le programme écrit au cours de cet examen pour résoudre la "douzième épreuve". Cette-fois-ci, les règles ont un peu changé :

- il y a maintenant 9 cellules ;
- une seule cellule contient une princesse, les autres sont soit vides soit contiennent un tigre ;
- l'affiche sur la cellule 1 dit: *"La princesse est dans une cellule dont le numéro est impair."*
- l'affiche sur la cellule 2 dit: *"Cette cellule est vide."*
- l'affiche sur la cellule 3 dit: *"L'affiche 5 est vraie ou l'affiche 7 est fausse."*
- l'affiche sur la cellule 4 dit: *"L'affiche 1 est fausse"*
- l'affiche sur la cellule 5 dit: *"L'affiche 2 ou l'affiche 4 est vraie"*
- l'affiche sur la cellule 6 dit: *"L'affiche 3 est fausse"*
- l'affiche sur la cellule 7 dit: *"La princesse n'est pas dans la cellule 1"*
- l'affiche sur la cellule 8 dit: *"Cette cellule contient un tigre et la cellule 9 est vide"*
- l'affiche sur la cellule 9 dit: *"Cette cellule contient un tigre et l'affiche 6 est fausse"*.
- l'indice donné par le roi est "l'affiche de la cellule de la princesse dit la vérité, celles des cellules qui contiennent un tigre mentent, pour les cellules vides on ne sait rien."

Il y a une difficulté supplémentaire dans cet exercice: une assertion ne parle plus d'un triplet, mais peut parler d'une autre assertion. Vous devrez enrichir votre programme en conséquence. Ecrivez le programme et faites le tourner sous clingo. Vous pourrez vérifier que le roi est un tricheur! Pourquoi?

Ayant démontré que le roi est un tricheur, le prisonnier pose une question : la cellule 8 est-elle vide? Lorsque le roi répond franchement, le prisonnier devine où est la princesse. Expliquez ceci en complétant votre programme.

Question 18 *(Je pense encore plus difficile, je n'ai pas trouvé de solution satisfaisante)* Si Raymond Smullyan était encore parmi nous, et qu'il jetait un coup d'oeuil à notre programme, il pourrait nous dire, comme le prisonnier de la question précédente: *Vous avez triché! Vous présentez ici une méta-logique du premier ordre, mais vos agents ne peuvent exprimer que du propositionnel !* Expliquez cette affirmation, en justifiant cette explication par des exemples bien choisis. Pouvez-vous proposer une ébauche de solution ?