

# Problemes ASP

baget

January 2021

## 1 Modélisations en ASP

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.

### 1.1 Meta-modélisation: les bases

**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  $\Pi_1$  le fait que si un agent dit quelque-chose, 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.

- `truth(C) :- say(A, C), not lie(C).`
- `lie(C) :- say(A, C), not truth(C)`
- `:- lie(C), truth(C)`

**Question 2** Ce programme  $\Pi_1$  est-il stratifiable? Justifiez votre réponse.

**Question 3** Le programme  $\Pi_2$  est obtenu à partir du programme  $\Pi_1$  en rajoutant les atomes `say(bob, c)`, `lie(c)`. Par la méthode vue en cours, transformez le programme  $\Pi_2$  en un programme propositionnel  $\Pi_3$ . **Lors du passage en propositionnel, 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  $\Pi_3$ , vous pourrez dans les questions qui suivent le remplacer par le programme suivant (qui n'est pas tout à fait  $\Pi_3$ ).*

- `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  $\{\text{sbc}, \text{lc}\}$  est un modèle stable de  $\Pi_3$ .

**Question 5** En justifiant de la manière la plus rapide possible (c'est à dire en évitant la méthode du point fixe), dire si  $\{\text{sbc}, \text{tc}\}$ ,  $\{\text{sbc}\}$  et  $\{\text{sbc}, \text{tc}, \text{lc}\}$  sont des modèles stables de  $\Pi_3$ .

**Question 6** Combien  $\Pi_3$  a-t'il de modèles stables? Pouvez-vous vous servir de la réponse à cette question pour répondre à la question 2?

## 1.2 Meta-modélisation: liaison avec le réel

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  $(\text{princess}, \text{in}, \text{cell1})$  est une notation alternative pour l'atome  $\text{in}(\text{princess}, \text{cell1})$  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.

En rajoutant les règles exprimant ces connaissances au programme  $\Pi_1$ , vous obtenez le programme  $\Pi_4$ .

- `triple(S, P, O):- claim(C, S, P, O), truth(C).`
- `:- claim(C, S, P, O), lie(C), triple(S, P, O).`
- `truth(C) :- claim(C, S, P, O), triple(S, P, O).`
- `lie(C) :- claim(C, S, P, O), not triple(S, P, O).`

**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  $\Pi_5$  suivant (où le symbole  $\neq$  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  $\neq$  E',  
not triple(E, in, C).`
- `:- triple(E, in, C), triple(E', in, C), E  $\neq$  E'.`

Déroulez l'algorithme ASPERIX (au brouillon) sur ce programme  $\Pi_5$  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  $\Pi_6$  composé des programmes  $\Pi_4$  et  $\Pi_5$ . On obtient le programme  $\Pi_7$  en y rajoutant les 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 cet exemple. Attention, certains arbres peuvent être beaucoup plus grands que d'autres!

### 1.3 Méta-modélisation: conjonction, négation, disjonction.

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** Nous obtenons le programme  $\Pi_8$  en ajoutant au programme  $\Pi_6$  le programme suivant (où `conjunction(C, C1, C2)` veut dire que `C` est la conjonction de `C1` et de `C2`):

- `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  $\Pi_8$  auquel 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?

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

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

## 1.4 Application: la première épreuve

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".

**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 connaissance par une assertion "say"), dit que "*l'une des 2 affiches dit la vérité, et l'autre ment*". Modélisez cette connaissance.

## 1.5 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 jusqu'ici, et que vous avez encore un peu de temps, les questions suivantes sont des "questions bonus". Elles ne sont pas comptabilisées dans le barème, 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](mailto: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*) 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.