Université de Montpellier – Master Informatique
Fondements de l'IA symbolique (HAI710I)

TP Moteur de règles (suite)

Nous utilisons à nouveau le moteur de règles ASP **Clingo**, disponible ici (ou en téléchargement) :

https://potassco.org/clingo/run/

Dans le TP précédent, nous avons considéré des bases de connaissances ayant **un seul** modèle stable (la notion de modèle stable correspondant à celle de base de faits saturée obtenue par une dérivation persistante et complète). Nous allons maintenant considérer des bases de connaissances ayant un nombre quelconque de modèles stables (**aucun** si la base de connaissances est insatisfiable, **un** seul si l'ensemble de règles est stratifiable, ou **plusieurs** dans le cas général).

En ASP, un modèle stable est appelé une réponse ("answer"). L'idée est que l'on peut décrire des problèmes sous forme de faits et règles, et chaque base de faits saturée formant une "réponse" correspond à une solution au problème.

Pour commencer : zéro, un ou plusieurs modèles stables ?

On considère la base de connaissances ci-dessous, où le prédicat frereOuSoeur se définit simplement par le fait d'avoir un parent en commun (règle R).

```
parent(a,c).
parent(a,d).
parent(a,e).
parent(b,c).
parent(b,d).
parent(b,e).

frereOuSoeur(X,Y) :- parent(Z,X), parent(Z,Y). % règle R

#show frereOuSoeur/2. % restriction affichage pour lisibilité
```

1) Exécuter cet exemple avec Clingo : on obtient un seul modèle stable ("`Models : 1`").

2) Cependant, certains individus se retrouvent frereouSoeur d'eux-mêmes, ce que l'on veut éviter. On pense à ajouter une contrainte négative interdisant d'avoir des faits de la forme frereouSoeur(X,X) :

```
:- frereOuSoeur(X,X). % On ne peut pas être frère ou soeur de soi-même
```

La base de connaissances devient insatisfiable ! Où est l'erreur ? Pour satisfaire la règle R et les faits, la base de faits saturée doit contenir des atomes de la forme frereOuSoeur(X,X). Par exemple, puisque qu'on a le fait `Parent(a,c)` et un homomorphisme {Z -> a, X -> c, Y -> c} du corps de R dans ce fait, le fait `frereOuSoeur(c,c)` doit appartenir à la base de faits saturée. Ceci viole la contrainte négative, on a donc une contradiction. On n'a donc plus *aucun* modèle.

3) C'est donc la règle qu'il faut modifier pour imposer que X soit différent de Y. Vérifier qu'on a toujours un seul modèle stable, et que la présence ou non de la contrainte ne change rien.

4) Pour avoir une base saturée plus petite, on voudrait maintenant éviter que pour deux individus i1 et i2 reconnus comme frereOuSoeur, on ait à la fois frereOuSoeur(i1,i2) et frereOuSoeur(i2,i1). On modifie donc

la règle ainsi : "si X et Y différents ont un parent en commun, et qu'on ne sait pas que Y est frère ou sœur de X, alors X est frère ou sœur de Y".

```
frereOuSoeur(X,Y) :- parent(Z,X), parent(Z,Y), X!=Y, not frereOuSoeur(Y,X).
```

Le résultat semble satisfaisant, cependant Clingo nous dit qu'il y a éventuellement plusieurs modèles stables ("`Models:1+`"). Remarquer au passage que l'ensemble {R} n'est pas stratifiable (boucle étiquetée négativement).

Pour voir tous les modèles stables, exécuter Clingo avec le mode de raisonnement "`enumerate all`" au lieu de "`default`" (par défaut, Clingo affiche seulement le premier modèle stable trouvé). Combien de solutions y a-t-il ? Voir que chacune de ces solutions est bien obtenue par une dérivation persistante.

Remarque : pour résoudre la question précédente et garder un seul modèle stable, on peut mettre un ordre total sur les constantes (en particulier c < d < e), et demander dans le corps de la règle que X soit plus petit que Y (X < Y). La règle redevient positive, ce qui assure l'unicité du modèle stable.

Dans les exercices suivants, vous ne devez utiliser que des règles de la forme `tête :- corps`, où la **tête** comporte **au plus un atome** - elle est vide dans le cas d'une contrainte négative - et le corps est une conjonction de littéraux.

Attention : la virgule correspond à un ET en corps de règle et à un OU en tête de règle (mais de toutes façons, on vous demande de ne pas l'utiliser en tête de règle).

Vous n'avez pas le droit d'utiliser les macros que fournit Clingo pour simplifier le code, sauf en ce qui concerne la macro suivante :

```
p(1..n). % p est un prédicat unaire
         % écriture équivalente à p(1). p(2). etc p(n).
```

Coloration de graphes

Nous nous intéressons maintenant à la modélisation de problèmes de satisfaction de contraintes. L'approche générale est la suivante :

- les faits décrivent les données du problème
  (par exemple : un graphe, peut-être aussi des couleurs)
- les règles permettent de générer des affectations
  (par exemple : affecter une couleur à tout sommet du graphe)
- les contraintes négatives décrivent les contraintes qui doivent être respectées
  (par exemple : deux sommets adjacents ne peuvent pas avoir la même couleur).

Clingo calcule tous les modèles stables de la base de connaissances, c'est-à-dire toutes les bases de faits saturées (par des dérivations persistantes) qui respectent les contraintes négatives, et chacune correspond à une solution.

1) Écrire une base de connaissances qui résout le problème suivant : étant donné un graphe, est-il colorable en 3 couleurs ?

Aide (d'autres modélisations sont possibles) :

- Base de faits : on décrit un graphe par ses sommets (prédicat unaire node) et ses arêtes (prédicat binaire edge). Par exemple : `node(1). node(2). node(3). edge(1,2). edge(2,3).`

- Base de règles :
    - On se donne un prédicat binaire liant un sommet à sa couleur : color(X,C) signifie que X est coloré par C
    - On déclare que tout sommet est coloré en rouge, en vert ou en bleu (constantes r,g,b). Ceci peut s'exprimer par des règles affirmant que si un sommet n'est pas coloré par deux des couleurs alors il est coloré par la troisième.
    - On impose que deux sommets adjacents soient forcément colorés par des couleurs différentes (contrainte négative).

Dans notre modélisation, on affirme qu'un sommet doit avoir *au moins une* couleur, mais on n'affirme pas qu'il doit en avoir *au plus une* : ce ne sera pas nécessaire, car nos règles ne chercheront pas à colorer un sommet avec deux couleurs. Si on voulait l'exprimer, on pourrait ajouter une contrainte négative interdisant qu'un sommet ait deux couleurs différentes.

Pour analyser le comportement de votre base de connaissances, vous pouvez pré-colorer certains sommets de façon à réduire le nombre de réponses possibles.

2) Résoudre le **problème de coloration de la carte de l'Australie** vu en cours/TD.

Puzzle du zèbre

Résoudre le puzzle du zèbre (cf. TD3). Il existe de multiples façons de modéliser ce problème : lorsque vous avez trouvé une modélisation qui fonctionne, essayez de la simplifier et de la rendre la plus claire possible.

Attention : vous n'avez pas droit aux macros que fournit Clingo, sauf celle donnée à la page précédente (encart jaune).

Pour l'affichage du résultat, définir par exemple un prédicat 6-aire qui fournit le descriptif de chaque maison sous cette forme : `maison(Numéro,Couleur,Nationalité,Boisson,Animal,Cigarette).` Ensuite, demander de restreindre l'affichage aux atomes ayant le prédicat maison.

Contrôle de TP (6/12)

Pour le contrôle de TP (sur machine), ayez résolu les exercices sur la famille d'Œdipe (TP Clingo 1), la coloration de l'Australie et si possible le puzzle du zèbre (celui-ci étant le plus difficile à modéliser).