ASP : Answer Set Programming

Carito Guziolowski
Département MATH-INFO, LS2N, École Centrale de Nantes

9 février, 2023

TD2 : ASP

Notions:

- Sémantique
- Écriture de Programmes Logiques en ASP

1 Interprétations et Modèles Herbrand Stables

Calculer les modèles Herbrand Stables (ou Answer Sets) du programme suivant :

```
man(dilbert).
single(X) :- man(X), not husband(X).
husband(X) :- man(X), not single(X).
```

2 Qui a tué Tante Agathe

Quelqu'un dans le manoir de Dreadsbury a tué Tante Agatha. Agatha, le majordome, et Charles vivent à Dreadsbury Mansion, et sont les seuls à y vivre. Un tueur déteste toujours sa victime, et il/elle n'est pas plus riche que sa victime. Charles ne déteste pas les personnes détestés par Agatha. Agatha déteste tout le monde sauf le majordome. Agatha ne déteste pas quelqu'un qui est détesté par le majordome. Le majordome déteste tout le monde qui n'est pas plus riche que Tante Agatha. Personne déteste tout le monde. Ecrire le programme logic en ASP pour trouver qui a tué Agatha.

3 Soirée du reveillon

Vous organisez une grande soirée du reveillon. Il y aura n tables, avec m chaises par table. Les invités seront numérotés de 1 à $m*n$. Vous avez besoin de choisir une table pour chaque invité avec les restrictions suivantes : (1) certains invités ont des affinités et voudriont s'assoir ensemble, et (2) certains invités ont des antipathies et ne voudriont pas s'assoir ensemble. L'instance de ce programme est composée par les prédicats binaires : `affinite(`k_1`,`k_2`)` et `antipathie(`t_1`,`t_2`)`, où k_i et t_i seront des invités. Écrivez les règles du programme logic pour garantir que chaque couple d'invités

I,J va s'assoir ensemble s'ils sont dans une relation de type *affinité* et les placer dans des tables différentes dans le cas où leur relation soit de type *antipathie*. L'objectif est d'attribuer à chaque invité une table en respectant son affinité/antipathie vis à vis d'autres invités, ou de déterminer que cette attribution est impossible.

4 Contraintes pour la coloration d'un graphe

Pour un graphe orienté et signé, nous disposons d'un coloriage partiel de ses noeuds (voir Figure 1). Les signes admisibles des arcs du graphe sont "$+$" et "$-$". Les couleurs admisibles pour les noeuds du graphe sont *rouge* (down, $-$), et *vert* (up, $+$).

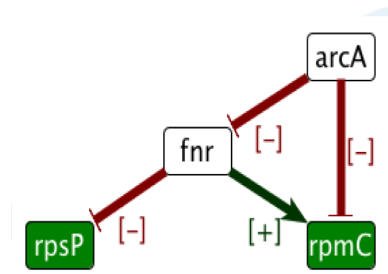

Figure 1: Graphe orienté et signé avec un coloriage partiel de ses noeuds.

Nous voudrions raisonner sur ce graphe, avec la règle de consistance suivante: *le signe (couleur vert ou rouge respectivement) d'un noeud avec prédecesseurs doit être égale au signe de l'influence d'au moins un de ses prédecesseurs directes dans le graphe.*

L'influence d'un noeud du graphe i sur son successeur j dans le graphe est égale au produit de son signe (*signe(*i*)*) et de l'arc sur j, i.e. *signe(*$edge(i,j)$*)*. Suivant ce raisonnement nous voyons dans la Figure 2 des exemples de graphes avec un coloriage inconsistante.

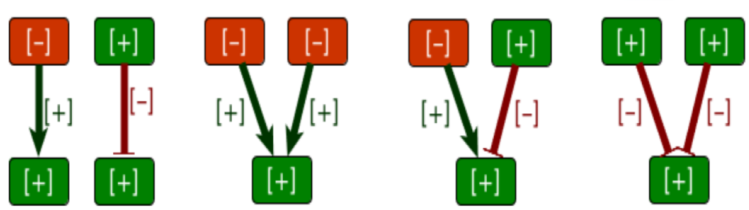

Figure 2: 5 graphes orientés et signés avec des coloriages inconsistantes.

4.1 Question 1

Écrire le programme logic en ASP qui permet effectuer le raisonnement suivant :

- Déduire si un graphe est consistant ou pas avec un coloriage partiel donné

- Si un graphe est consistant avec un coloriage partiel, alors nous voudrions obtenir tous les modèles de coloration totale consistantes. Dans un modèle de coloration totale, chaque noeud du graphe doit avoir une couleur rouge ou vert associée.

Utilisez comme instance du problème le graphe avec le coloriage partiel donné dans la Figure 1.

5 Optimal Golomb Ruler

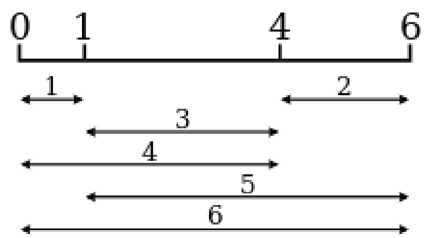

Figure 3: Règle de Golomb de longueur minimale 6 avec 4 marques

Écrire un PL pour placer un nombre donné de marques sur une règle, telle que deux paires de marques ne soient jamais à la même distance. En d'autres termes chaque couple de marques mesure une longueur différente des autres couples. La première marque doit être placée à la position 0, et la dernière marque doit être placée à la longueur de la règle. La longueur de la règle doit être minimale[1]. On impose finalement que la distance entre un pair de marques soit un nombre L fixé par le prédicat `lenghtNeeded/1`. Imaginez que l'instance de ce problème a la forme suivante :

```
position(0..50). mark(1..8). lengthNeeded(16).
```

[1]Ce problème a des applications dans la conception des antennes, et la technologie de communication mobile