Lab 2:
Prolog, logic programming, constraint programming

INF8215 – Intelligence artificielle : méthodes et algorithmes

Auteurs:
Alexandre Boudreault (1486776)
Arjun Gupta (1965475)

11 Novembre 2018

Exercice 1

Voici les résultats qui satisfont les contraintes de notre modèle MiniZinc :

5	japanese	acrobat	zebra	green	coffee
4	spanish	violinist	dog	white	orangeJuice
3	english	sculpter	snails	red	milk
2	ukranian	physician	horse	blue	tea
1	norwegian	diplomat	fox	yellow	water

Chaque rangée du tableau représente une maison ayant une position, une nationalité, une profession, un animal de compagnie, une couleur et une boisson.

La position représente la position de la maison dans le voisinage, «1» étant la maison la plus à gauche et «5» étant la maison la plus à droite.

Question 1.1 : Qui boit de l'eau?

Le norvégien boit de l'eau.

Question 1.2 : Qui possède le zèbre?

Le japonais possède un zèbre.

Exercice 2

Dans cet exercice, il fut question de déterminer l'ordre des parties. Nous avons implémenté deux méthodes avec les mêmes résultats, mais des vitesses d'exécution différentes.

```
3   7   2   9   5   4   10  11  6   8   13  12  14
7   8   1   6   4   3   11  9   12  5   10  14  13
1   6   4   5   7   2   9   8   10  13  14  11  12
6   5   3   7   2   1   8   12  13  14  9   10  11
8   4   6   3   1   11  12  13  14  2   7   9   10
4   3   5   2   11  10  13  14  1   7   12  8   9
2   1   9   4   3   12  14  10  11  6   5   13  8
5   2   12  13  10  14  4   3   9   1   11  6   7
11  10  7   1   14  13  3   2   8   12  4   5   6
12  9   13  14  8   6   1   7   3   11  2   4   5
9   13  14  12  6   5   2   1   7   10  8   3   4
10  14  8   11  13  7   5   4   2   9   6   1   3
14  11  10  8   12  9   6   5   4   3   1   7   2
13  12  11  10  9   8   7   6   5   4   3   2   1
----------
Finished in 483msec
```

Pour lire ce tableau, il faut comprendre que chaque représente une équipe, chaque colonne représente une ronde et chaque valeur associée à une colonne et une rangée données représente l'équipe adverse.

Dans ce premier cas, nous avons dû s'assurer que si une équipe jouait contre une autre, l'autre équipe devait nécessairement jouer contre la première. Ceci fut implémenté à l'aide de deux contraintes:

- `matches[t2,m] == t1 -> matches[t1,m] == t2`
- `matches[t1,m] == t2 -> matches[t2,m] == t1`

De plus, nous devons ajouter une contrainte qui assure que si une équipe joue «home», l'équipe adverse doit nécessairement jouer «away». Encore une fois, nous avons utilisé deux contraintes chaque:

- `pv[t1,t2] == 0 -> pv[t2,t1] == 1`
- `pv[t1,t2] == 1 -> pv[t2,t1] == 0`

On remarque que le programme s'exécute en 483 msec. Pour accélérer cet exécution, nous avons combiner les paires de contraintes pour diminuer le nombre de calculs. Plutôt que de dire «si X, alors Y» et puis «si Y, alors X», nous avons fait appel à la symétrie du problème pour tout simplement dire «si et seulement si X, alors Y».

- `matches[t1,m] == t2 <-> matches[t2,m] == t1`
- `pv[t1,t2] == 1 <-> pv[t2,t1] == 0`

Le tableau suivant présente ce qui est obtenu avec ces petites modifications. Les résultats sont les mêmes, mais le temps d'exécution est significativement moins élevé.

```
3   7   2   9   5   4   10  11  6   8   13  12  14
7   8   1   6   4   3   11  9   12  5   10  14  13
1   6   4   5   7   2   9   8   10  13  14  11  12
6   5   3   7   2   1   8   12  13  14  9   10  11
8   4   6   3   1   11  12  13  14  2   7   9   10
4   3   5   2   11  10  13  14  1   7   12  8   9
2   1   9   4   3   12  14  10  11  6   5   13  8
5   2   12  13  10  14  4   3   9   1   11  6   7
11  10  7   1   14  13  3   2   8   12  4   5   6
12  9   13  14  8   6   1   7   3   11  2   4   5
9   13  14  12  6   5   2   1   7   10  8   3   4
10  14  8   11  13  7   5   4   2   9   6   1   3
14  11  10  8   12  9   6   5   4   3   1   7   2
13  12  11  10  9   8   7   6   5   4   3   2   1
----------
Finished in 305msec
```

Bonus

Le tableau suivant présente quelles parties sont joués «home». Chaque rangée représente une équipe, chaque colonne une ronde et chaque élément représente un booléen indiquant si la partie est joué «home».

```
1   1   0   0   0   1   0   1   0   0   1   1   1
0   1   1   0   1   1   0   1   1   1   1   1   0
0   0   0   1   0   0   0   1   1   1   0   0   1
0   0   1   1   0   0   1   1   1   0   0   0   1
1   1   1   0   1   0   0   1   0   0   0   1   0
1   1   0   1   1   0   0   0   1   1   1   1   1
1   0   0   0   1   0   1   1   1   0   1   1   0
0   0   0   1   0   0   0   0   0   1   0   0   1
1   1   1   1   0   1   1   0   1   0   1   0   0
1   0   0   1   1   1   1   0   0   0   0   1   1
0   0   0   0   0   1   1   0   0   1   1   1   0
0   0   1   1   0   1   1   0   0   1   0   0   0
0   1   1   0   1   0   1   0   0   0   0   0   1
1   1   1   0   1   1   0   1   1   1   1   0   0
```

On remarque ici que l'équipe 2 joue 5 parties «home» de suite. Disons maintenant que l'on veut ajouter une contrainte où une équipe peut jouer un maximum de 4 parties «home» en ligne. Pour ce faire, nous ajoutons la contrainte suivante:

```
constraint forall(t in 1..nbTeams, m in 1..(nMatches - 3))(
     pv[t, matches[t, (m+0)]] == 0 \/
     pv[t, matches[t, (m+1)]] == 0 \/
     pv[t, matches[t, (m+2)]] == 0 \/
     pv[t, matches[t, (m+3)]] == 0
);
```

Cette contrainte stipule que pour chaque groupe de 4 matchs joués par une équipe, au moins une doit être «away». Le tableau suivant présente les résultats obtenus.

```
2         3         6         5         9         4         7         11        8         10        13        12        14
1         7         12        3         4         6         8         9         5         11        10        14        13
6         1         4         2         5         13        9         8         10        7         14        11        12
7         6         3         8         2         1         14        12        13        5         9         10        11
12        14        11        1         3         8         6         13        2         4         7         9         10
3         4         1         13        7         2         5         10        11        14        12        8         9
4         2         10        11        6         9         1         14        12        3         5         13        8
13        10        9         4         14        5         2         3         1         12        11        6         7
10        11        8         12        1         7         3         2         14        13        4         5         6
9         8         7         14        13        11        12        6         3         1         2         4         5
14        9         5         7         12        10        13        1         6         2         8         3         4
5         13        2         9         11        14        10        4         7         8         6         1         3
8         12        14        6         10        3         11        5         4         9         1         7         2
11        5         13        10        8         12        4         7         9         6         3         2         1

0         1         0         0         0         1         1         1         0         0         1         1         1
1         0         1         1         1         0         1         1         1         0         1         1         0
0         0         0         0         1         1         0         1         1         0         0         0         1
1         0         1         1         0         0         0         1         1         0         0         0         1
0         0         0         1         0         1         1         1         0         1         0         1         0
1         1         1         0         1         1         0         0         1         0         1         1         1
0         1         1         1         0         0         0         1         0         1         1         1         0
1         0         0         0         0         0         0         0         1         0         0         0         1
1         1         1         0         1         1         1         0         0         1         1         0         0
0         1         0         1         0         0         1         1         0         1         0         1         1
0         0         1         0         0         1         0         0         0         1         1         1         0
1         0         0         1         1         0         0         0         1         1         0         0         0
0         1         0         1         1         0         1         0         0         0         0         0         1
1         1         1         0         1         1         1         0         1         1         1         0         0
----------
Finished in 340msec
```

Ayant les même conventions que les tableaux présentés plus haut, la partie supérieur présente les adversaire de chaque équipe lors de chaque match alors que la partie inférieure présente si ce match est joué «home».

On peut remarquer que les matchs diffèrent légèrement du résultat précédent, mais nous ne retrouvons plus de séries de 4 matchs «home» de suite. L'exécution est légèrement plus lente, mais elle tombe à l'intérieur de la moyenne des exécutions précédentes de la solution optimisée.

Exercice 3

Voici une session prolog illustrant différents outputs de `completeRequirementsFor/2` pour trois cours différents:

```
[?- [question3].
 true.

[?- completeRequirementsFor('INF1500',L).
 L = [].

[?- completeRequirementsFor('INF1900',L).
 L = ['INF1005C', 'INF1500', 'INF1600', 'INF2205', 'LOG1000'].

[?- completeRequirementsFor('INF2705',L).
 L = ['INF1005C', 'INF1010', 'INF2010', 'LOG2810', 'LOG2990', 'MTH1007'].
```

Les cours sont représentés par des objets String entre guillemets de manière à ce qu'ils ne soient pas considérés comme des variables du fait que leur première lettre soit une majuscule.

Nous pouvons aussi vérifier les corequis et prérequis de chaque cours. Un cours corequis est par définition considéré comme prérequis aussi (mais pas l'inverse). Les corequis sont symétriques comme on peut le voir ci-dessous.
On écrit "X est un prérequis pour le cours Y" dans prolog comme: `prerequisite(x,y).`
On écrit "X est un corequis pour le cours Y" dans prolog comme: `corequisite(x,y).`

```
[?- corequisite('LOG1000','INF1900').
true .

[?- corequisite('INF1900','LOG1000').
true.

[?- prerequisite('LOG1000','INF1900').
true.
```

Un cours non reconnu par la base de connaissance renverra comme output "false". Un cours sans prérequis renverra une liste vide. Ce programme est conçu uniquement pour les cours principaux que l'on voit dans l'arbre donné par l'énoncé.

Exercice 4

Voici un exemple du programme de devinette avec deux personnes et un objet en tête.

```
?- [question4].
true.

?- person(X).
Is the person a leader? yes.
Is the person a political leader? |: yes.
Is the person in power in 1953? |: no.
Is the person russian? |: no.
Is the person american? |: no.
Is the person egyptian? |: yes.

X = cleopatra .

?- person(X).
Is the person a leader? no.
Is the person fictional? |: no.
Is the person related to Hollywood? |: yes.
Is the person a movie director? |: no.
Is the person an actress? |: no.
Is the person an actor? |: yes.

X = denzel_washington .

?- object(X).
Does the object use electricity? yes.
Is the object used for making food or beverages? |: no.
Is the object used to clean things? |: no.
Does the object have a screen? |: yes.
Does the object fit in a pocket? |: no.

X = computer .

?-
```

Puisque le programme se résoud de façon binaire et pose uniquement des questions avec pour réponse “oui” ou “non”, l’ajout d’une nouvelle personne ou d’un nouvel objet implique l’ajout d’au moins une question pour pouvoir départager le litéral ajouté aux autres litéraux existant. Par exemple si nous voulions ajouter l’objet “knife”, nous pourrions ajouter le litéral `cutlery(knife)` à la base de connaissance et ajouter une question pour différentier un `cutlery(knife)` d’un `cutlery(fork)`.