INF8215

Intelligence artificielle : méthodes et algorithmes

TP2

Prolog, programmation logique, programmation par contraintes

Par :

Nathan Heraief - 1973553

Paul-Arthur Thiéry - 1969108

Jean-Raphaël Cornel - 1978029

Département de génie informatique

Polytechnique de Montréal

11 novembre 2018

Exercice 1

Dans cet exercice, nous traduisons toutes les affirmations dans miniZinc. Il suffit ensuite de résoudre pour avoir le résultat demandé.
Le résultat est exprimé avec un chiffre, car à chaque personne est associée un chiffre.
En l'occurrence, pour les deux questions posées, c'est le Norvégien qui boit de l'eau et le Japonais qui possède un zèbre.

Exercice 2

Ici, nous décidons de représenter notre calendrier comme une matrice. Chaque ligne de la matrice est un jour du calendrier, chaque colonne est une équipe participant au tournoi. Les éléments individuels sont les équipes rencontrées par l'équipe de la colonne où l'on se trouve. Il nous suffit donc de mettre des contraintes de différences par ligne et par colonne afin d'avoir une calendrier cohérent. Ensuite, nous ajoutons une contrainte liée à pv pour calculer si les 4 derniers matchs étaient à domicile, à l'extérieur ou non.

En implémentant seulement ces éléments simples, nous obtenons une solution en 1 minute et quelques secondes. Toutefois, si nous utilisons la symétrie des matchs (si l'équipe A affronte l'équipe B alors l'équipe B affronte l'équipe A), nous avons un résultat en environ 300msec. Cela est dû à la division du travail à accomplir, car sur chaque ligne seulement la moitié des matchs doivent être créés, et ce nombre diminue au fur et à mesure que la matrice se remplit grâce à nos contraintes de différences en ligne et colonne.

Exercice 3

On décide de simplifier les liens entre les cours avant de les implémenter en Prolog. Pour cela, on peut d'abord regrouper tous les cours corequis en une seul classe d'équivalence. Ainsi, pour avoir les cours corequis d'un autre cours, il suffit de regarder quels sont les autres cours dans la classe d'équivalence. On implémente ainsi la règle corequis, qui est par nature symétrique.
On redéfinit donc le lien de prérequis entre les classes d'équivalence de cours. Une classe est prérequise à une autre si l'un de ses cours au minimum est prérequis à au moins un cours de l'autre classe d'équivalence. On implémente directement les prérequis d'une classe à l'autre.

Prédicats:

- cours(x): x est un cours
- prerequis(x1,x2): Le cours x2 est un prérequis du cours x1
- classe(y): y est une classe
- classeAppartenance(y, x): le cours x appartient a la classe d'équivalence y
- classePrerequis(y1,y2): La classe y2 est un prérequis de la classe y1

Règles:

- corequis(C1, C2): Vrai si les cours C1 et C2 sont corequis; c'est à dire s'ils appartiennent à la même classe et sont différents
- prerequisComplet(C1, C2): Vrai si les cours C1 et C2 sont corequis, ou si C2 est un prérequis direct de C1
- classeRequis(Classe, C): Vrai si le cours C appartient à classe, où si C appartient à une classe parente de Classe de manière récursive.
- toutRequis(C1, C2): vrai si les cours C1 et C2 sont corequis, ou si C2 appartient à une des classes prérequises de la classe de C1 de manière récursive.
- courAPrendreComplet(C, L): Vrai si L est la liste triée de l'ensemble des cours prérequis ou corequis du cours C.

Exercice 4

Il s'agit ici de reproduire le modèle du très connu site web : Akinator. On veut qu'à l'aide de devinette, notre algorithme trouve l'objet ou la personne auquel pense le joueur.

Il faut donc séparer l'exercice en 2 parties : l'une pour les personnes et l'autre pour les objets.

Chaque partie est donc structurée entre la base de connaissance et le schéma d'implication. Les schéma d'implication qui sont le coeur de l'algorithme sont détaillés ci dessous :

Schéma Personne(X). :

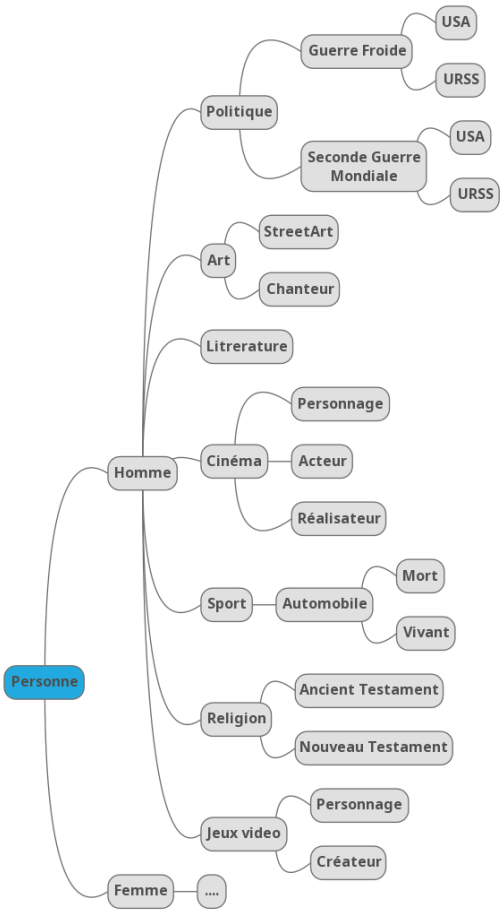

Schéma Objet(X). :