T. P. Programmation par contraintes.
POLYTECH. 4ième année. Alternance
UNIVERSITÉ D'AIX-MARSEILLE
Année universitaire 2018/2019

Enseignant : Odile Papini

**Feuille de T. P. 3 : Résolution de CSP en PROLOG avec Gnu Prolog**

**Utilisation de Gnu Prolog :**

- Le site web de Gnu Prolog : *http* : *//www.gprolog.org/*
- La documentation de Gnu Prolog : *http* : *//www.gprolog.org/♯manual*

Pour utiliser Gnu Prolog :

- création (avec un éditeur quelconque) d'un fichier d'extension .pl qui contient le programme.
- compilation (2 manières possibles) :
  - gplc nom_prog.pl
  - gplc -o nom_prog_exe nom_prog.pl

  *Dans le premier cas, l'exécutable a le même nom que le programme source, dans le deuxième cas, il peut avoir un nom différent.*
- execution
  - ./ nom_prog_exe

  *affichage d'une nouvelle invite |?-*
  *il est alors possible d'interroger le programme*
- quelques commandes : **a** : abort, **c** : continue, **d** : debug, **h** : help, **b** : break, **e** : exit, **t** : trace.

**Description de CSP binaire**

Limitation aux contraintes portant sur 2 variables exactement. La description d'un CSP binaire en PROLOG se fait comme suit :

- Pour représenter les variables et les domaines, construction d'une liste $L$ qui représente les variables et pour chaque variable les valeurs du domaine, telle qu'un élément de $L$ est de la forme : le nom de la variable suivi du symbole : suivi de la liste des valeurs du domaine soit : $nom\_de\_variable_i$ : $[di_1, \ldots, di_m]$

  ATTENTION le nom des variables et les valeurs du domaine doivent être des constantes.

- Définition d'un fait qui spécifie que la liste $L$ représente les variables et les valeurs des domaines pour le probème CSP. Utiliser le prédicat unaire $variables(L)$.

- Pour représenter les contraintes, définition de règles qui représentent les conditions qui entraînent la vérification des contraintes. La tête de règle $consistants((X_i, V_i), (X_j, V_j))$ spécifie que l'affectation partielle $\{(X_i, V_i), (X_j, V_j)\}$ est consistante si la(ou les) contrainte(s) binaire(s) entre $X_i$, et $X_j$ du problème CSP est (sont) vérifiée(s) pour les valeurs $V_i$ et $V_j$.

**Exercice 1 : Description du problème des 4 reines**

Ecrire un programme en Gnu-Prolog qui

- Décrit les variables de ce problème .
- Décrit les contraintes de ce problème.

Tester le programme pour obtenir la liste des variables et des valeurs des domaines, pour vérifier que les contraintes entre deux variables sont vérifiées ou pas.

**Résolution d'un CSP binaire avec l'algorithme génère et teste**

- Pour générer toutes affectations possibles des variables aux valeurs des domaines, parcourir la liste qui représente les variables et les valeurs des domaines pour le problme CSP et construire au fur et à mesure

du parcours une nouvelle liste qui contient les affections des variables aux valeurs des domaines. Ecrire les règles avec le prédicat binaire $genere(L, L')$

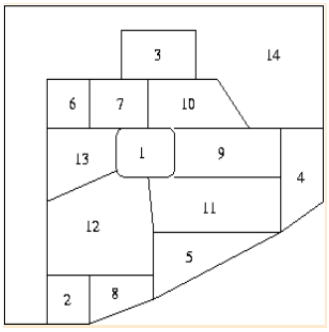
- Pour tester si les affectations satisfont les contraintes, parcourir la liste des affectations et vérifer si l'affectation courante $(X_i, V_i)$ de la liste satisfait les contraintes avec des éléments de la queue de la liste. Ecrire les règles avec le prédicat binaire $teste(L)$.
- Pour vérifier qu'une affectation $(X_i, V_i)$ de la liste satisfait les contraintes avec des éléments de la queue de la liste. Ecrire les règles avec le prédicat binaire $verifie((X, V), L)$.

**Exercice 2 : Résolution du problème des $n$ reines avec génère et teste**

1) Reprendre la description du problème de l'Exerxice 1.
2) Ecrire un programme qui résoud le problème avec l'algorithme génère et teste.

**Exercice 3 : Coloriage de carte**

Il s'agit de colorier les 14 régions de la carte ci-dessous, de sorte que deux régions ayant une frontière en commun soient coloriées avec des couleurs différentes. On dispose pour cela des 4 couleurs suivantes : bleu, rouge, jaune et vert.




1) Modéliser ce problème sous la forme d'un CSP.
2) Donner la description en Prolog du problème de coloriage de carte.

3) Ecrire un programme qui résoud le problème avec l'algorithme génère et teste.

**Exercice** 4 **: Sel + moutarde**

1) Donner la description en Prolog du problème.

2) Ecrire un programme qui résoud le problème avec l'algorithme génère et teste.