PRLOG : Programmation Logique

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

15 février 2018

TD3 : ASP

Notions :
— Écriture de Programmes Logiques en ASP
— Type de problèmes de recherche combinatoire : puzzles, concepts formels, planification.

1 Cryptarithme

Selon Wikipedia *un cryptarithme, aussi connu sous les noms d'arithmétique verbale, d'alphamétique et de cryptarithmétique, est un casse-tête numérique et logique qui consiste en une équation mathématique où les lettres représentent des chiffres à trouver*. Un des cryptarithmes classique est celui décrit en 1924 par Henry Dudeney :

$$\begin{array}{rr} & \text{SEND} \\ + & \text{MORE} \\ \hline & \text{MONEY} \end{array}$$

Écrire un programme logique (PL) en ASP pour trouver à quoi corresponde le terme $MONEY$ dans ce cryptarithme.

2 Concepts formels

Contexte : nous pouvons à ce jour, grâce à des technologies récentes, mesurer la quantité des protéines *actives* chez un individu. Nous pouvons alors construire de matrices booléennes M de ce type de données de taille $m \times n$ pour m individus et n protéines, où m_{ij} représente le niveau (actif ou absent) de la protéine j pour l'individu i. Ces individus, lorsqu'ils sont atteints de maladies comme le cancer, peuvent être traités. Il y a cependant des individus qui vont bien réagir après un traitement et d'autres que non. Selon ce critère ils peuvent être classés en 2 classes différentes : CR (*complete remission*, bon pronostic), et PR (*primary resistant*, mauvais pronostic). Une façon de comprendre les causes de cette différence est de construire des graphes avec un sous-ensemble de ces protéines et pour un sous-ensemble de ces individus. Dans ce processus de construction de graphes de protéines, un filtre de ce type des matrices es nécessaire.
Problème : Concevoir un programme logique en ASP que étant donnée une matrice booléenne de taille $m \times n$ permet de construire une sous-matrice en assemblant quelques colonnes et quelques lignes selon les critères suivants :

1. Sélectionner un sous-ensemble $\mathcal{K}$ composé de k protéines, à partir de toutes les combinaisons possibles sur les n protéines.
2. Sélectionner des pairs des individus pour lesquels les valeurs booléens des k protéines choisisses coïncident pour les 2 classes (CR et PR).
3. Maximiser le nombre des pairs des individus appartenant à des classes différentes (voir 1).

$$\begin{array}{ll} \text{maximiser} & \displaystyle\sum_{\mathrm{j,j'} \in \mathrm{CR} \times \mathrm{PR}} \mathrm{f^K(i, i')} \\ \text{sujet à} & f^K(i, i') = 1 \text{ si } m_{ij} = m_{i',j} \forall j \in \mathcal{K} \\ & f^K(i, i') = 0 \text{ sinon.} \end{array} \tag{1}$$

L'instance de ce problème peut être composée par les prédicats `protein/1` (les protéines), `class/1` (les classes), et `exp(I,J,S,C)`, qui exprime $m_{IJ} = S$ (la valeur de la protéine J de l'individu I est S, $S \in \{0, 1\}$) et que l'individu I appartient à la classe C.

Listing 1 – Instance du problème

```
1 protein(1..3).
2 class(c1).        class(c2).
3 exp(1,1,0,c1).    exp(1,1,1,c1). exp(1,1,0,c1).
4 exp(2,1,0,c2).    exp(2,1,1,c2). exp(2,1,0,c2).
5 ...
```

3 Tours de Hanoï

Le problème des tours de Hanoï consiste à déplacer des disques de diamètres différents d'une tour de *départ* à une tour de *arrivée* en passant par une tour *intermédiaire*, et ceci en un minimum d'étapes, tout en respectant les règles suivantes :

— on ne peut déplacer plus d'un disque à la fois,

— on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

On suppose que cette dernière règle est également respectée dans la configuration de départ. Dans la Figure 1 nous montrons la configuration de départ d'une instance de ce problème.

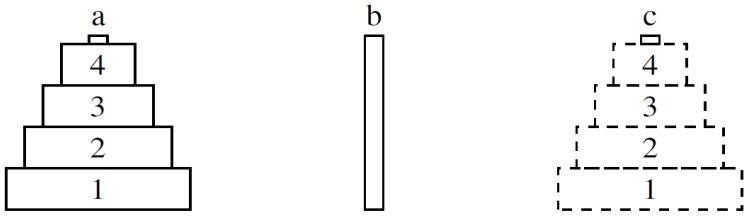

FIGURE 1 – Configuration de départ pour le problème de Tours de Hanoï.

Ces faits peuvent être représentés avec les prédicats suivants :

```
1 tour(a;b;c).
2 disque(1..4).
3 conf_init(1..4,a).
4 conf_objectif(1..4,c).
5 deplacements(15).
```

Écrire le PL en ASP qui permet de répondre si on peut passer de la configuration initiale décrite par le prédicat `conf_init` à la configuration finale (`conf_objectif`) avec 15 déplacements, et d'afficher les déplacements à faire dans le cas de satisfiabilité. Un déplacement peut être représenté par le prédicat `deplacer(D,T,M)` qui modélise le fait de deplacer le disque D dans la tour T dans le déplacement numéro M. Utiliser la méthode *Génerer, Définir et Tester* pour écrire le PL.