Toolbox Intelligence Artificielle

L’objectif de cet exercice est l’écriture d’un système expert permettant d’empiler des cubes dans l’ordre croissant comme dans la figure 1. L’objectif est de réussir l’empilement malgré de potentiels aléas comme l’ajout, la suppression ou la modification des cubes en cours d’exécution

1

3

6

94

51

12

état initial : état final :

94

1

SOL

51

6

3

12

SOL

# Partie 1 : Implémentation Java et premières règles.

Une première étude naïve du problème donne la modélisation suivante : si pour chaque cube, je connais sa valeur, son identifiant et l’identifiant du cube sur lequel il repose (le sol sera considéré comme une constante) alors il suffira de tout mettre au sol puis d’empiler dans l’ordre voulu.

**Question 1.1** : *Utiliser l’API Jess (http://www.jessrules.com/jess/docs/71/api/index.html)*

Utiliser la classe *Cube.java* définit dans le projet [*http://www.emse.fr/~picard/cours/ai/#sec-4-5*](http://www.emse.fr/~picard/cours/ai/#sec-4-5)*.* Ecrire la classe *Main.java* pour créer les cubes et les ajouter dans la base de faits de l’objet *Rete* que vous aurez créé.

**Question 1.2**: Ecrire dans le fichier *cube.jess* les règles suivantes :

* + une règle pour déplacer un cube sur le sol (qui peut en accepter autant qu’il en faut).
  + une règle pour déplacer le cube sur celui qui a sa valeur juste inférieure.

***Remarque*** : attention aux contraintes physiques, par exemple, un cube ne peut être déplacé que s’il est libre (au début seuls les cube 1 et 94 de la figure précédente peuvent être déplacés). Pour chaque déplacement réalisé, la partie action de la règle permettra d’afficher le cube grâce à la méthode *afficher* de Cube.

**Question 1.3** : Compléter la classe *Main* pour exécuter et afficher la base de faits.

**Question 1.4** : Si le résultat n’est pas satisfaisant modifier la modélisation précédente.

# Partie 2 : Intégration des perturbations

La classe *Perturbation.java* du projet génère aléatoirement des listes ordonnées correspondant aux trois types de perturbations proposés.

**Question 2.1** : Ajouter une liste ordonnée *(time <timeValue>)* lors de l’exécution du système expert qui sera modifiée (*timeValue* est diminuée de 1) à chaque déclenchement d’une règle.

**Question 2.2** : La règle *hasard* se déclenche dès que la valeur 0 est atteinte et réinitialise la valeur à la valeur de la variable globale TIMEVALUE. La partie action de la règle modifie la base de faits selon la perturbation générée. Enfin une nouvelle perturbation est générée.

**Question 2.3** : Proposer une base de règles permettant de continuer l’empilement malgré la perturbation.

# Complément

Rete r = new Rete();

Maclasse mc = new MaClasse();

r.defclass("objet", "package.Maclasse", null);

r.definstance("objet",mc,false);