Systèmes à base de connaissances - CLIPS

L'objectif de cette séance de TME est de manipuler le logiciel CLIPS pour concevoir et utiliser un système à base de connaissances (SBC).

Compte-rendu : avant la séance suivante, vous rendrez le fichier CLIPS de l'exercice 3 "Diagnostic médical" en précisant, en commentaires, la maladie trouvée, la suite des règles déclenchées et les faits ajoutés.

Exercice 1 *Prise en main de CLIPS*

Pour commencer, télécharger le fichier "tme3.tgz" sur le Moodle de l'UE. Cette archive contient un répertoire avec 2 fichiers clp qui sont des fichiers clips.

Ensuite, lire l'annexe de ce sujet qui présente les commandes de base du logiciel CLIPS et lancer CLIPS (commande "clips" dans un terminal de commandes). Pour cet exercice, charger le fichier `famille.clp` dans CLIPS par la commande `(load "famille.clp")`.

Question 1. Représenter graphiquement sur une feuille la hiérarchie familiale correspondant à l'ensemble des faits du fichier.

Question 2. Essayer les commandes de base de l'interpréteur CLIPS et expliquer leur résultat :

```
CLIPS> (reset)
CLIPS> (rules)
CLIPS> (facts)
CLIPS> (agenda)
CLIPS> (run)
```

Question 3. Tester d'autres commandes de l'interpréteur CLIPS : (watch facts), (watch rules), (clear). Par exemple :

```
CLIPS> (reset)
CLIPS> (watch facts)
CLIPS> (watch rules)
CLIPS> (run)
```

A chaque étape, noter la règle déclenchée et donner l'état de la base de faits.

Question 4. Modifier la base de faits initiale afin de tester différentes possibilités d'inférence avec ce système à base de connaissances.

Question 5. Que se passe-t-il si

- on retire le fait 0, juste après `(reset)` et avant de lancer les inférences ? Pourquoi ?
- on effectue la séquence suivante :

  ```
  CLIPS>(clear)
  CLIPS>(rules)
  CLIPS>(facts)
  CLIPS>(reset)
  CLIPS>(facts)
  ```

  Que peut on en déduire sur le rôle du fait 0 ?

Question 6. Recopier le fichier "famille.clp" pour créer le fichier "famille-suite.clp" dans lequel vous rajouterez une règle "oncle_tante" permettant de créer le fait correspondant à la propriété "`?enf` a pour oncle ou pour tante `?x`". On considèrera la définition réduite : un oncle ou une tante est le frère ou la sœur d'un parent.

Question 7. Rajouter dans le fichier précédent une règle "cousin_cousine" permettant de créer le fait correspondant à la propriété "`?enf1` a pour cousin ou cousine `?enf2`".

Question 8. Recopier le fichier "famille.clp" pour créer le fichier "famille-unique.clp" dans lequel vous rajouterez la règle suivante :

```
(defrule enfant_unique_bad
        (parent ?enf ?papaOuMaman)
        (not (frere_et_soeur ?enf ?autre))
=>
        (assert (enfant_unique_bad ?enf))
)
```

Ensuite, charger cette base et lancer les inférences. Le résultat obtenu est-il correct ? Quelle explication pouvez-vous donner ?

Question 9. Remplacer la règle précédente par la règle suivante :

```
(defrule enfant_unique_priorite
        (declare (salience -1000))
        (parent ?enf ?papaOuMaman)
        (not (frere_et_soeur ?enf ?autre))
=>
        (assert (enfant_unique_priorite ?enf))
)
```

Puis, charger et exécuter cette base. Qu'en conclure ?

Question 10. Proposer une autre solution pour maintenir une base de faits cohérente sans priorités.

Exercice 2 *Gestion d'une cave à vins*

Pour cet exercice, vous devez relancer clips et charger le fichier "wine.clp".

Question 1. Essayer les commandes de base :

```
CLIPS> (reset)
CLIPS> (facts)
CLIPS> (run)
```

Question 2. Tester d'autres commandes de l'interpréteur CLIPS : (watch facts), (watch rules), (clear), (agenda). Par exemple :

```
CLIPS> (reset)
CLIPS> (watch facts)
CLIPS> (watch rules)
CLIPS> (run)
```

A chaque étape, noter la règle déclenchée et donner l'état de la base de faits.

Question 3. Modifier la base de faits initiale afin de tester différentes possibilités d'inférence avec ce système à base de connaissances.

Exercice 3 *Diagnostic médical*

Le but de cet exercice est de construire un système à base de connaissances (SBC), sur le modèle du système MYCIN, capable de diagnostiquer certaines maladies.

On s'intéresse ici à des maladies infantiles comme la varicelle, la rougeole... Voici les connaissances à utiliser :

- si le sujet a peu ou beaucoup de boutons on dit qu'il a comme symptôme une éruption cutanée.
- on dit que le sujet a un exanthème s'il a des éruptions cutanées ou des rougeurs.
- un sujet est dans un état fébrile s'il a une forte fièvre ou s'il ressent une sensation de froid.
- le fait d'avoir les amygdales rouges, des tâches rouges et la peau qui pèle est un signe suspect.
- la rougeole est diagnostiquée si le patient est dans un état fébrile et qu'il a des yeux douloureux et un exanthème, ou bien s'il a un signe suspect et une forte fièvre.
- s'il a peu de fièvre et peu de boutons, ce n'est pas la rougeole.
- on relève une douleur si le patient a les yeux ou le dos douloureux.
- la grippe est diagnostiquée si le sujet a le dos douloureux et un état fébrile.
- on peut diagnostiquer la rubéole et la varicelle si on a déjà déterminé qu'il ne s'agissait pas de la rougeole.

— si le sujet a de fortes démangeaisons et des pustules alors c'est la varicelle.
— s'il a la peau sèche, une inflammation des ganglions mais ni pustules ni sensation de froid, c'est la rubéole.

Question 1. Implémenter en CLIPS ce SBC pour réaliser un diagnostic médical.

Question 2. On considère les faits initiaux suivants :
— le patient a des tâches rouges ;
— il a peu de boutons ;
— il ressent une sensation de froid ;
— il a une forte fièvre ;
— il a mal aux yeux ;
— ses amygdales sont rouges ;
— sa peau pèle et elle est sèche.

A l'aide du SBC implémenté, étudier les inférences réalisées à partir de ces faits. En particulier, observer la trace des règles appliquées `(watch rules)` et des faits déduits `(watch facts)` puis lancer n itérations `(run` n`)` (avec n un entier naturel qui donne le nombre maximum de règles à déclencher, au plus) en regardant à chaque étape `(agenda)`.

Question 3. Déterminer quelle est la stratégie de CLIPS pour choisir la règle à activer quand plusieurs règles sont applicables.

Question 4. Clips procède-t-il par chaînage avant ou par chaînage arrière ?

Question 5. Saturer la base de faits `(run)` pour déterminer tout ce qui peut être prouvé.

Question 6. Repérer la maladie diagnostiquée et, à la main, faire un chaînage arrière. Comparer le nombre de règles utilisées avec le nombre de règles utilisées par CLIPS.

Annexe : le logiciel CLIPS

Sur les machines en salle de TME, le logiciel se lance dans un terminal par la commande `clips` :

```
~$ clips
         CLIPS (6.30 3/17/15)
CLIPS>
```

Pour plus d'informations voir : `http://www.clipsrules.net/Documentation.html`

Commandes de base CLIPS est un interpréteur de commandes et les commandes sont donc entrées en mode interactif. On peut écrire une base de règles dans un fichier (cf. la syntaxe du langage décrit ci-après) afin de l'utiliser dans plusieurs sessions CLIPS.
— charger un fichier : utiliser la commande `(load <nom_du_fichier>)`
— initialiser la base de faits : `(reset)`
— lancer le moteur d'inférences (saturer la base de règles à partir de faits initiaux donnés) : `(run)`
— quitter CLIPS : `(exit)`
— les commentaires dans un fichier sont identifiés par le caractère "`;`" en début de ligne.

Faits
— insérer un fait dans une base de faits :
`(assert (duck))` : insère le fait duck dans la base de faits courante.
`(assert (a) (b) (c 0 1))` : insère plusieurs faits simultanément (3 dans cet exemple).
remarque : Si le fait inséré existe déjà dans la base de faits, alors le système retourne `false`.
— enlever un fait de la base de faits : `(retract 1)` : enlève le fait d'indice 1 de la base de faits.
— lister les faits de la base de faits courante : `(facts)`
`(facts 4)` : affiche tous les faits depuis celui d'indice 4.
— supprimer tous les faits et toutes les règles : `(clear)`

— afficher au fur et mesure les ajouts et les retraits de la base de faits `(mode trace)` :
`(watch facts)` et `(unwatch facts)`
`==>` (resp. `<==`) symbolise l'ajout (resp. le retrait) d'un fait de la base de faits.
— préparer un ensemble de faits initiaux à ajouter à la base de faits grâce la commande `reset` :
`(deffacts nom_de_la_base_de_faits fait1 fait2 ...)`
— préparer un ensemble de faits initiaux, à ajouter à la base de faits après exécution du `run` :

```
(defrule initialisation "initialisation des faits"
  (initial-fact)
  =>
  (assert fait1 fait2 ...))
```

Variables

— une variable s'écrit : `?<nom_de_la_variable>`
remarque : la variable "`?`" correspond une variable anonyme.
— affecter le fait `(fait1)` à la variable `?x` : `?x <- (fait1)`
— affecter la valeur `val` à la variable `?x` : `(bind ?x val)`

Règles

— une règle est définie de la manière suivante :

```
(defrule nomDeLaRègle "commentaires"
    (pattern1)
    ...
    =>
    (action1)
    .... )
```

— afficher la définition d'une règle :
`(ppdefrule nomDeLaRègle)` ou `(ppdefrule NomDuModule::nomDeLaRègle)`
— avoir le nom de toutes les règles présentes dans la base de règles : `(rules)`
— voir les déclenchements de règles : `(watch rules)`
— quelques exemples de patterns de conditions :
  — existence d'un fait : `(nomDuPrédicat param_1 param_2 ...)`
  — test sur les variables : `(test (or...))` `(test (and...))`

```
    (test (eq...)) ; pour les chaînes de caractères
    (test (=...)) ; pour les nombres
```

  — négation d'une condition : `(not condition)`
— quelques exemples d'actions :
  — examen des règles : `(watch rules)`
  — ajout d'un fait : `(assert (nomDuPrédicat param_1 param_2...))`
  — retrait d'un fait : `(retract ind_f1 ind_f2)`
  — affichage : `(printout t chaîne_de_caractères crlf)`

Structures de contrôle comme actions

— l'alternative :

```
(if condition then
    action1
    ...
    else
    action1
    ...)
```

— l'itérative :

```
(while condition
    action1
    ...)
```