Rapport de IA01 :
Intelligence artificielle – Représentation des connaissances

UNIVERSITE DE TECHNOLOGIE DE COMPIEGNE

Automne 2016

Guillaume JOUNEL & Julien JERPHANION

Sujet du rapport :
TP03 : Réalisation d'un système expert d'ordre 0+

Département des étudiants :
Génie Informatique

Professeur :
Marie-Hélène ABEL

Table des matières

Liste des programmes

1 Introduction : Présentation du système expert

1.1 Problématique

Tous les programmeurs sont un jour confrontés au problème suivant :

> « *Quels de programmation et technologies sont les plus adaptés pour le projet que je souhaite développer dans mon cadre d'utilisation ?* »

Pour pallier à ce problème, nous allons concevoir un système expert qui propose différentes possibilités les plus adaptées selon l'usage.

Pour cela, nous prendrons en compte de multiples critères tels que le domaine d'application (calcul numérique, intelligence artificielle...), l'expérience de l'utilisateur ou encore les caractéristiques de sa machine (Linux, MacOS...).

1.2 Sources de connaissances sur le sujet

Les sources d'expertise ne manquent pas : il existe de nombreux sites et ressources sur le Net qui donnent les avantages et inconvénients de tous les langages de programmation existants selon les cas d'utilisation. En voici quelques uns :

- Wikipédia : Liste des langages de programmations par type : `https://en.wikipedia.org/wiki/List_of_programming_languages_by_type` ;
- Learneroo : The Different Programming Languages : `https://www.learneroo.com/modules/12/nodes/94` ;
- WhoIsHostingThis : What Code Should You Learn ? `http://www.whoishostingthis.com/blog/2014/09/04/learn-to-code/`.

2 Architecture

2.1 Rappels sur l'architecture

Rappelons rapidement l'architecture d'un système expert. Un système expert est constitué de trois parties principales dissociées les unes des autres : une *base de faits*, une *base de règles*, et un *moteur d'inférences*.

La base de faits est une base d'informations qui comprend les faits initiaux mais aussi déduits au cours du programme.

La base de règles contient les différentes règles (connaissances implicites de l'expert rendues explicites pour être représentées informatiquement) utilisées pour déduire d'autres faits.

Les inférences au cours du processus sont réalisées par le moteur d'inférences. C'est lui qui fait le lien entre les deux précédentes bases. Il exécute les règles contenues dans la base de règles au regard des faits présents dans la base de faits ; les règles étant déclenchables en fonction des faits avérés. À la fait de l'exécution d'une règle, le résultat retourné qui est aussi un fait est stocké dans la base de faits.

Il existe de type de fonctionnement pour les moteurs d'inférences : le *chaînage avant* et le *chaînage arrière*.

Le chaînage avant consiste à regarder les faits présents et à choisir une règle qui peut être exécutée : on cherche les résultats que l'on peut obtenir en se basant sur les résultats déjà obtenus.

Le chaînage arrière examine les règles à exécuter pour arriver à un certain fait : on cherche un moyen d'arriver à un certain résultat.

2.2 Base de faits

Nous avons implémenté nos faits selon la forme suivante :

```
(objet EQ valeur)
```

La base de faits est stockée dans une variable globale `*faits*` initialement vide : elle se remplira au cours de l'exécution du système.

2.3 Base de règles

Nous avons décidé d'implémenter notre base de règle de cette façon :

```
(((Premisse1 operateur valeur)...(PremisseN operateur valeur))
     ((Resultat1 operateur valeur)...(Resultat1M operateur valeur)))
```

```
(defparameter *regles* '(
  ; Regles construites de la fonctions suivantes :
  ; (((Premisse1 operateur valeur)...(PremisseN operateur valeur))
  ;    ((Resultat1 operateur valeur)...(Resultat1M operateur valeur)))

  ; Site web
  (((Application EQ Site-Web-Simple))
    ((Propositions EQ (PHP MySQL))))
  (((Application EQ Site-Web-Simple))
    ((Propositions EQ (PHP MySQL))))
  (((Application EQ Site-Web-Responsive))
    ((Propositions EQ (PHP MySQL BootStrap JavaScript))))
  (((Application EQ Site-Web-Efficace))
    ((Propositions EQ (Django))))
  (((Application EQ Site-Web-Efficace))
    ((Propositions EQ (Ruby-on-Rails))))
  (((Application EQ Site-Web-Simple))
    ((Propositions EQ (PHP MySQL))))
  (((Application EQ API))
    ((Propositions EQ (Django Django-Rest-Framework))))

  ; Application Mobile
  (((Application EQ Mobile)(Machine EQ Mac)(Cible EQ iPhone) (Budget > 100))
    ((Propositions EQ (Swift))))
  (((Application EQ Mobile) (Cible EQ Android))
    ((Propositions EQ (JAVA Android-Studio SDK-Android))))
  (((Application EQ Mobile) (Cible EQ Android))
    ((Propositions EQ (JAVA Android-Studio SDK-Android))))

  ; Logiciel
  (((Application EQ Logiciel) (Precision EQ Solide))
    ((Propositions EQ (C++ JAVA))))
  (((Application EQ Logiciel) (Precision EQ Solide) (Machine EQ Windows))
    ((Propositions EQ (C#))))

  ; Calcul-Numerique
  (((Application EQ Calcul-Numerique) (Utilisation Individuel) (Budget > 100))
    ((Propositions EQ (Matlab))))
  (((Application EQ Calcul-Numerique) (Budget < 100))
    ((Propositions EQ (Scilab Julia ))))

  ; Système-expert
  (((Application EQ Systeme-Expert) (Parenthese EQ Supportee))
    ((Propositions EQ (LISP))))
  (((Application EQ Systeme-Expert) (Parenthese EQ Non-Supportee))
    ((Propositions EQ (Prolog))))

  ; Applet
  (((Application EQ Mobile) (Machine EQ Mac) (Cible EQ Mac))
    ((Propositions EQ (Swift))))
  (((Application EQ Applet) (Usage EQ Personnel))
    ((Propositions EQ (Pygame Tkinter))))

  ; Jeu-Video
```

2.4 Base de connaissances

Pour donner les informations concernant les technologies choisies par *Cactus* nous avons décider d'implémenter une base de connaissances. Celle-ci contient pour chaque technologie une brève description de celle-ci.

Les éléments des cette base sont représentés sous la forme de liste pointée ainsi :

```
(technologie . "La description de la technologie")
```

```
(defparameter *technologies*
  '(
    (C . "Un des langages les plus populaires et très bien structuré ; utilisé pour accéder à la mémoire de la machine, pour la création de système d'exploitation.")
    (C++ . "Inspiré du C, il en reprend beaucoup de spécificités et est orienté programmation objet.")
      (QtScript . "A COMPLETER")
    (Python . "Langage interprété de prototypage. Efficace et très simple d'utilisation ; de nombreuses bibliothèques.")
      (Tkinter . "Bibliothèque Python : permet de créer des interfaces graphiques.")
      (PyGame . "Bibliothèque Python : permet de créer des petits jeux et applications graphiques.")
      (Django . "FrameWork de développement web Python : très bien construit et extremement efficace une fois maîtrisé 'Django : the web framework for perfectionists with deadlines'.")
      (Numpy-MathPlotLib . "Bibliothèque python : boîte à outils scientifiques")
      (Sci-kit . "Bibliothèque python : boîte à outils scientifiques")
    (Ruby . "Un langage polyvalent qui 'rend les developpeur heureux'. Il est proche des langages comme Python mais est surtout utilisé pour le developpement web avec Ruby-on-Rails")
      (Ruby-on-Rails . "Le framework Ruby pour le developpement web. Plus populaire que Django.")
    (R . "Utilisé dans les domaines scientifiques (particulièrement en statistiques et data-mining)")
    (MatLab . "Le langage reconnu pour ces fonctionalité en analyse numérique et calcul scientifique. Possède de très nombreuses fonctionalités mais est payant.")
    (Octave . "Logiciel et langage de programmation de calcul numérique, alternative libre et gratuite à Matlab.")
    (LISP . "((((((((((((((((((((LE FAMEUX))))))))))))))))))))")
    (Scheme . "A COMPLETER")
    (Pascal . "Un langage de programmation ancien; sa synthaxe est simple ce qui lui donne un bon atout pédagogique.")
    (Prolog . "Un langage de programmation français (Cocorico !), il est utilisé en Intelligence Artificielle.")
    (Scala . "acronyme de ''Scalable Language'' ; langage péblicité pour des applications nécessitant de gérer de nombreuses tâches en parallèles.")
    (SQL . "Le standard des bases de données relationnelles : c'est à la fois un langage et une technologies d'implémenation de BDD ; connait plusieurs variantes d'implémentations")
      (PL/SQL . "Variante propriétaire d'Oracle ; permet d'implémenter des bases de données relationnelles-objet et dispose de fonctions et fonctionnalités supplémentaires.")
      (PostGreSQL . "Variantes libre d'implémentation de SQL la plus populaire.")
      (MySQL . "Variantes libre d'implémentation de SQL, simple d'utilisation mais aussi moins bien structurée.")
    (VBA . "Permet de créer des applications légères dans Excel pour traiter des petits ensemble de données ; solution accessible aux non developpeurs mais payante.")
    (Arduino . "Cartes électroniques de 'hacking' en license libre. Elles se
```

Nous utiliserons cette base de connaissances dans la fonction `afficherPropositions` que nous détaillerons plus bas : l'idée est d'avoir un petit descriptif des technologies proposées pour comprendre en quoi elles sont pertinentes.

3 Fonctionnement du système

3.1 Chaînage avant

Voici un algorithme itératif pour le chaînage avant.

Algorithme 1 : Chaînage Avant

début
  faire
    pour chaque règle r **de la base de règle** BR **faire**
      si r **est déclenchable alors**
        $EC \leftarrow EC \cup \{r\}$;
        $BR \leftarrow BR - \{r\}$;
      si $EC \neq \emptyset$ **alors**
        $BF \leftarrow BF \cup$ **conclusions**(r);
      sinon
        poser une question
      ;
  tant que il n'y a pas de propositions dans BF;
  afficher les propositions ;

On remarquera que le système s'arrêtera dès que des propositions auront été inférées par le moteur.

Nous l'avons implémenté sous LISP de cette façon :

```lisp
(defun chainageAvantLarg ()
  ;Moteur chaînage avant en largeur
  (let (EC regleCourante)
    (loop ; on boucle
      (if (valeur (assoc 'Propositions *faits*)) ; si le but est présent dans la base de faits avec une valeur non nulle
          (progn
            (funcall 'afficherPropositions)
            (return nil))
          (dolist (r *regles*) ; sinon on parcourt les règles dans la base de règles
            (when (declenchable? r); si une règle est déclenchable
              (setq EC (append EC (list r))) ; on l'ajoute à l'ensemble contraint EN FIN
              (setq *regles* (remove r *regles* :test 'equal))))) ; on l'enlève de la base de règles
      (if EC ; si on peut encore déclencher des règles
          (progn
            (setq regleCourante (pop EC)) ; on choisit la dernière obtenue
            (ajouter (conclusion regleCourante))) ; on ajoute son résultat à la base de faits
          (askQuestion)))))
```

Programme 3: Moteur de chainage avant

3.2 Fonctions outils

```
;;Fonctions outils pour les règles
(defun conclusion (r)
  (cadr r))

(defun declenchable? (r)
  (let (
      (OK t)
      (premisses (car r))
    )
    (dolist (p premisses OK)
      (if (and (numberp (valeur p)) (not (valeur (assoc (objet p) *faits*))))
        (setq OK nil)
        (if (not (funcall (operateur p) (valeur (assoc (objet p) *faits*))
↪  (valeur p)))
          (setq OK nil))))))

(defun ajouter (resultats)
  ; Ajoute un résultat à la base de faits *faits*
  ; fait : triplet de la forme (objet opérateur valeur)
  (dolist (triplet resultats)
    (if (assoc (objet triplet) *faits*) ; si l'objet est déjà présent dans la
↪  base
      (setf (caddr (assoc (objet triplet) *faits*)) (caddr triplet)) ; on
↪  remplace sa valeurs
      (push triplet *faits*)))) ; sinon on rajoute triplet à la base de faits
```

Programme 4: Fonctions outils pour les règles

```
;; Fonctions outils pour les faits (triplets)
(defun objet (triplet)
  (car triplet))

(defun operateur (triplet)
  (cadr triplet))

(defun valeur (triplet)
  (caddr triplet))
```

Programme 5: Fonctions outils pour les faits

3.3 Poser une question : fonction `askQuestion`

```
(defun listeAttFaits ()
  (loop for fait in *faits*
    collect (objet fait)
  )
)

(defun listeAttRegles ()
  (loop for regle in *regles*
    append (loop for premisse in (car regle)
      collect (objet premisse)
    )
  )
)

(defun AttValues (attribut)
  (loop for regle in *regles*
    if (assoc attribut (car regle))
    collect (valeur (assoc attribut (car regle)))
  )
)

(defun askQuestion ()
  (let ((attribut (car (set-difference (listeAttRegles) (listeAttFaits)))
  ↪ valeur))
    ;"attribut" est le premier élément de la différence entre :
    ; - la liste des attributs dans la base de faits
    ; - la liste des attributs dans la base de règles (prémisses)
    ;C'est-à-dire un attribut dont la valeur est inconnue.
    (if attribut
      (if (numberp (car (AttValues attribut)))
        (until
          (AND
            (not (format t "Spécifiez : ~S~&Votre choix (nombre) : "
  ↪ attribut))
            (numberp (setq valeur (read)))
          ) ;Redemande tant que son choix n'est pas valide
        )
        (until
          (AND
            ;liste les valeurs possibles de l'attribut et fait lire un choix
  ↪ à l'utilisateur
            (not (format t "Spécifiez : ~S~&~S~%Votre choix : " attribut
  ↪ (delete-duplicates (AttValues attribut))))
            (member (setq valeur (read)) (delete-duplicates (AttValues
  ↪ attribut)))
            ) ;Redemande tant que son choix n'est pas valide
        )
      )
      (error "Sorry, something went wrong")
    )
    (pushnew (list attribut 'EQ valeur) *faits*)
    ;  ajouter l'attribut / valeur à la base de regles
```

3.4 Afficher les propositions du système : fonction `afficherPropositions`

C'est dans cette fonction que nous utilisons la base de connaissances `**technologies**`.

```
1 (defun afficherPropositions ()
2   (let ((prop (caddr (assoc 'Propositions *faits*))) (overlay
  ↪  ------------------))
3     (format t "~&~A~&Voici les différentes technologies que je vous propose :
  ↪  " overlay)
4     (dolist (x prop)
5       (format t "~& -» ~A : ~S" x (cdr (assoc x *technologies*))))
6     (format t "~&~A~&" overlay)))
```

Programme 7: Fonction `afficherPropositions` qui affiche les propositions du système expert

⋆ ⋆ ⋆

Conduite d'expertise d'un SE d'ordre 0+

Dates de remise :

- **Lundi 28 novembre 2016 à 18H pour la réponse à la question 1.**
- **Lundi 9 janvier 2017 à 18H pour la réponse aux questions 2 et 3.**
- **Démonstration et présentation orale lors du dernier TD.**

L'objet du TP03 est de réaliser le développement d'un SE de sa phase d'expertise à sa phase d'utilisation. A cette fin, vous devez :

1 - Formalisez une problématique d'un domaine au choix (un qui vous passionne) qui puisse être traitée par un SE d'ordre 0+. Justifiez votre choix et faites-le valider par votre chargé de TD.
2 - Déterminez les connaissances nécessaires au SE : explicitez votre base de règles (donnez vos sources). Présentez l'arbre de déduction associé et donnez des jeux d'essais.
3 - Programmez votre SE
    a. Justifiez la représentation Lisp choisie pour exploiter les faits et la base de règles.
    b. Développez, justifiez et commentez le moteur d'inférences choisi : chaînage avant (ou arrière) en profondeur d'abord, chaînage avant (ou arrière) en largeur d'abord.
    c. Testez votre moteur et commentez les résultats. Une comparaison avec un deuxième moteur développé serait un plus.

Documents à produire :

- Un rapport écrit comportant les réponses aux points précédents et présentant des scénarios d'utilisation.
- Un fichier comportant le code lisp de votre SE avec les scénarios d'utilisation (à envoyer par courriel).
- Une courte présentation orale s'appuyant sur des transparents et une démonstration sont attendues au cours du dernier TD.