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

1 Introduction : Présentation du système expert **4**
1.1 Problématique 4
1.2 Sources de connaissances sur le sujet 4
1.3 Exemple d'intéraction 4
1.4 Exemples de règles 4

2 Architecture **5**
2.1 Base de règles 5

3 Base de faits & Bases de connaissances **7**

4 Fonctions outils **11**

Liste des programmes

1 Base de règles `*regles*` 6
2 Base de connaissances `*technologies*` 8
3 Différentes fonctions outils 11

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/`.

1.3 Exemple d'intéraction

Voici un exemple d'intéraction que l'on pourrait imaginer entre l'utilisateur et notre système :

SYSTÈME: Spécificiez : Catégorie
UTILISATEUR: [Application] – Calcul scientifique – Représentation de données
SYSTÈME: Spécificiez : Type d'application
UTILISATEUR: Système – Web – Jeu vidéo – DIY – [Applet] – Application Smart-Phone – Système Expert
SYSTÈME: Spécificiez : Usage
UTILISATEUR: [Personnel] – Réduit – Universel
SYSTÈME: Spécificiez : Niveau en Python
UTILISATEUR: Excellent – [Bon] – Aucun
SYSTÈME: Dans ce cas je vous conseille d'utiliser :

- **Python avec Tkinter** : permet de créer des interfaces graphiques (bibliothèque `Tk` ; plus d'information *ici* ;
- **Python avec Pygame** : adapté pour faire des petits jeux ; plus d'information *ici* ;

1.4 Exemples de règles

Voici quelques idées pour de règles que l'on pourrait implémenter (`Technologies` est liste des propositions de solutions proposées en résultats par le système expert) :

1. `SI (Catégorie = Application) ET (Type-application = Applet) ET (Usage = Personnel) et (Niveau-Python = Bon) ALORS (Technologies = (Tkinter Pygame))`
2. `SI (Catégorie = Application) ET (Type-application = Web) ET (Type-application-web = Statique) ET (Design = Aucun) ALORS (Technologies = (HTML5))`
3. `SI (Catégorie = Application) ET (Type-application = Système-Expert) ET (Intérêt = Pedagogique) ALORS (Technologies = (LISP Prolog))`

2 Architecture

2.1 Base de règles

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

```
(defparameter *regles* '(
        ; Règles construites de la fonctions suivantes :
        ;
        ; (((Premisse1 opérateur valeur)...(PremisseN opérateur
↪  valeur))
        ;          ((Résultat1 opérateur valeur)...(Résultat1M
↪  opérateur valeur)))

        ;; Affichage s'il y a des propositions
        ;;; Propositions Finales
        (((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 Mobile) (Machine EQ Mac))
                ((Propositions EQ (Swift)))) ;TODO : Choisir 1
        (((Application EQ Mobile)(Machine EQ Mac)(Cible EQ iPhone)
↪  (Budget > 100))
                ((Propositions EQ (Swift)))) ;TODO : Choisir 2
        (((Application EQ Mobile) (Cible EQ Android))
                ((Propositions EQ (JAVA Android-Studio
↪  SDK-Android))))
        (((Application EQ Logiciel) (Precision EQ Solide))
                ((Propositions EQ (C++ JAVA))))
        (((Application EQ Logiciel) (Precision EQ Solide) (Machine EQ
↪  Windows))
                ((Propositions EQ (C#))))
        (((Application EQ Systeme-Expert) (Parenthèse EQ Supportee))
                ((Propositions EQ (LISP))))
        (((Application EQ Systeme-Expert) (Parenthèse EQ
↪  Non-Supportee))
                ((Propositions EQ (Prolog))))
        (((Application EQ Applet) (Usage EQ Personnel))
                ((Propositions EQ (Pygame Tkinter))))
        (((Application EQ Systeme) (Machine EQ Linux) (Precision EQ
↪  Interaction))
                ((Propositions EQ (C Shell Tkinter))))
        (((Application EQ Jeu-Vidéo) (Precision EQ 3D))
                ((Propositions EQ (C++ Unity3D))))
        (((Application EQ DIY))
                ((Propositions EQ (Arduino))))
        (((Application EQ DIY) (Precision EQ Internet))
                ((Propositions EQ (Raspberry-Pi))))
        (((Application EQ Systeme-Embarque))
                ((Propositions EQ (Assembleur))))
))
```


Programme 1: Base de règles `*regles*`

3 Base de faits & Bases de connaissances

Nous avons implémenté nos faits selon la forme suivante :

```
(objet opérateur valeur)
```

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
```

Nous utiliserons cette base de connaissances dans la fonction `afficherPropositions` que nous détaillerons plus bas.

4 Fonctions outils

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

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

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

(defun operateur (triplet)
        (cadr triplet))

(defun valeur (triplet)
        (caddr triplet))

(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))))))


; (case (operateur p)
;         ('EQ
;                 (if (not (eq (valeur p) (valeur (assoc (objet p)
↪  *faits*))))
;                         (setq OK nil)))
;         ('>
;                 (if (not (> (valeur p) (valeur (assoc (objet p)
↪  *faits*))))
;                 (setq OK nil)))
;         ('>=
;                 (if (not (>= (valeur p) (valeur (assoc (objet p)
↪  *faits*))))
;                         (setq OK nil)))
;         ('<
;                 (if (not (< (valeur p) (valeur (assoc (objet p)
```

⋆ ⋆ ⋆