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 et 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 fin 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.

Il s'agit ici de construire un système expert d'ordre 0+ – que nous avons choisi d'appeler *Cactus* –, c'est à dire un système expert manipulant des faits qui ne sont non pas des propositions booléennes mais des *triplets* comportant trois parties :

1. un objet, qui est le nom du concept que l'on veut modéliser dans le fait ;
2. une valeur, qui permet de quantifier l'objet ;
3. un opérateur, qui permet de préciser la valeur de l'objet

Ainsi `(temperature >= 30)` et `(saison EQ été)` sont des faits vus sous l'angle d'un système-expert d'ordre 0+ dont les objets, les opérateurs et les valeurs sont respectivement `temperature` et `saison`, `>=` et `EQ`, et `30` et `été`.

2.2 Base de faits

Puisqu'il s'agit de concevoir un système expert d'ordre 0+, nous avons choisi d'implémenter 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. Il s'agira d'une liste de triplets.

Voici quelques objets que nous utiliserons pour modéliser les faits :

Objets	Signification
`Application`	le type d'application à développer
`Machine`	le type d'OS utilisé pour développer
`Cible`	le type d'OS visé pour l'application
`Budget`	le budget du développeur
`Precision`	pour préciser l'utilisation
`Utilisation`	précise le type d'utilisation

TABLE 1 – Exemple d'objets pour les faits

`Propositions` sera l'objet du faits utilisé pour stocker les différentes propositions inférées par *Cactus*.

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)))
```

où `operateur` $\in \{$`=`, `<`, `>`, `EQ`$\}$.

```
(defparameter *regles* '(

  ; 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 Individuelle) (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))))

```

```
51    ; Jeu-Video
52    (((Application EQ Jeu-Video) (Precision EQ 3D))
53      ((Propositions EQ (C++ Unity3D OpenGL))))
54    (((Application EQ Jeu-Video) (Precision EQ RPG-2D))
55      ((Propositions EQ (RPG-Maker))))
56
57    ;DIY
58    (((Application EQ DIY))
59      ((Propositions EQ (Arduino))))
60    (((Application EQ DIY) (Precision EQ Internet))
61      ((Propositions EQ (Raspberry-Pi))))
62
63    ;Système Embarqué
64    (((Application EQ Systeme) (Machine EQ Linux) (Precision EQ Interaction))
65      ((Propositions EQ (C Shell Tkinter))))
66    (((Application EQ Systeme-Embarque))
67      ((Propositions EQ (Assembleur Shell C))))
68
69    ;Autre
70    (((Application EQ RegEx))
71      ((Propositions EQ (Perl JavaScript))))
72
73    (((Application EQ Apprentissage))
74      ((Propositions EQ (Perl))))
75  ))
```

Programme 1: Base de règles `*regles*`

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.")
```


Programme 2: Base de connaissances `*technologies*`

```
27    (Arduino . "Cartes électroniques de 'hacking' en license libre. Elles se
      ↪ programment généralement dans un formalisme proche du C et du C++")
28    (HTML . "Formalisme de représentation de données utilisée par les pages web.
      ↪ La base du développement web")
29      (JavaScript . "Langage de programmation permettant de faire fonctionner
      ↪ des applications web côté client.")
30      (Json . "Un formalisme récent de représentation simple et lisible
      ↪ d'informations ; standard utilisé par beaucoup de langages.")
31      (XML . "Un formalisme plus ancien de représentation simple et lisible
      ↪ d'informations ; standard utilisé par beaucoup de langages.")
32    (Swift . "Le dernier langage de programmation d'Apple pour développer des
      ↪ applications iPhone et Mac")
33    (Objective-C . "L'ancien langage de programmation d'Apple pour développer
      ↪ des applications iPhone et Mac")
34    (C# . "Le langage de programmation orientée objet de Microsoft. Il révèle
      ↪ tout son potentiel s'il est utilisé conjointement au framework .NET.")
35    (Java . "Le langage de programmation orientée objet")
36      (NodeJS . "Plate-forme construite sur le moteur d'exécution JavaScript de
      ↪ Google Chrome. Permet de créer facilement des applications réseau rapides,
      ↪ évolutives et scalables.")
37      (AJAX . "pour ''Asynchronous JAvascript and Xml''. Architecture qui permet
      ↪ de créer des applications dynamiques.")
38    (Neo4J . "Technologie NoSQL : données représentées sous forme de graphes.")
39    (MongoDB . "Technologie NoSQL : données non structurées stockées sous le
      ↪ formalisme JSON")
40    (Cansadra . "Technologie NoSQL : A COMPLETER")
41    (PHP . "Le langage de developpement web le plus utilisé.")
42      (Symfony . "Sûrement le framework PHP le plus populaire : formalisme
      ↪ Modèle-Vue-Contrôleur")
43    (Fortran . "Vieux langage de programmation utilisé pour le calcul
      ↪ scientifique")
44    (Scilab . "Logiciel et langage de programmation de calcul numérique,
      ↪ alternative libre et gratuite à Matlab.")
45    (Julia . "Logiciel et langage de programmation de calcul numérique,
      ↪ alternative récente et libre à Matlab. Performant.")
46    (LaTeX . "Langage et un système de composition de documents. Utilisé pour la
      ↪ rédaction de documents scientifique. Beaucoup de bibliothèque (package)")
47    (Go . "Langage de programmation de Google ; se veut efficace et simple
      ↪ d'apprentissage.")
48    (Maple . "Logiciel de calcul formel ; propriétaire et payant.")
49    (Sage . "Se veut être ''une alternative viable libre et open source à Magma,
      ↪ Maple, Mathematica et Matlab''.")
50    (Perl . "Un langage de programmation simple à apprendre. Utile pour
      ↪ déterminer des expressions régulières")
51    (LolCat . "LOL U RLY NEEDIT")
52    (Piet . "ou l'art de coder.")
53    (Shell . "Interpréteur de commandes : l'interface entre l'OS et
      ↪ l'utilisateur")
54    )
55  )
```

Programme 3: Base de connaissances `*technologies*`

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

Comme nous l'avons évoqué au début, il existe plusieurs moyens de réaliser le moteur d'inférence : le chaînage avant et le chaînage arrière. De même pour chacune de ces façons de procéder, on peut choisir de parcourir l'arbre de déduction en profondeur ou en largeur.

3.1 Chaînage avant

Le chaînage avant a pour avantage d'être facilement implémentable et ne repose pas sur la recherche d'une réponse particulière contrairement au chaînage arrière dont l'algorithme peut se construire sur la recherche d'un but spécifique (comme nous avons pu le voir en TD dans le cas du chaînage arrière en profondeur d'abord avec les fonctions `verifier()` et `verifierET()`).

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 ← EC ∪ {r};
                BR ← BR − {r};
            si EC ≠ ∅ alors
                BF ← BF ∪ conclusions(r);
            sinon
                poser une question;
    tant que il n'y a pas de propositions dans BF;
    afficher les propositions ;
```

Cet algorithme permet de poser des questions à l'utilisateur lorsque le système ne peut plus inférer. L'utilisateur sera amené a préciser des valeurs d'objets pour que le moteur puisse ainsi construire de nouveaux faits. On remarquera que le système s'arrêtera dès que des propositions auront été inférées par le moteur.

3.1.1 En largeur

Nous avons tout d'abord réalisé un moteur d'inférence en chaînage avant et en profondeur d'abord.

Nous l'avons implémenté sous LISP de cette façon :

```
1  (defun chainageAvantLarg () ; Moteur chaînage avant en largeur
2    (let (EC regleCourante)
3      (loop ; on boucle
4        (if (valeur (assoc 'Propositions *faits*)) ; si le but est présent dans
   ↪  la base de faits avec une valeur non nulle
5          (progn
6            (funcall 'afficherPropositions) ; on affiche les propositions
7            (return nil)) ; et on arrête
8          (dolist (r *regles*) ; sinon on parcourt les règles dans la base de
   ↪  règles
9            (when (declenchable? r); si une règle est déclenchable
10             (setq EC (append EC (list r))) ; on l'ajoute à l'ensemble
   ↪  contraint EN FIN
11             (setq *regles* (remove r *regles* :test 'equal))))) ; on l'enlève
   ↪  de la base de règles
12       (if EC ; si on peut encore déclencher des règles
13         (progn
14           (setq regleCourante (pop EC)) ; on choisit la dernière obtenue
15           (ajouter (conclusion regleCourante))) ; on ajoute son résultat à la
   ↪  base de faits
16         (askQuestion))))) ; sinon on pose une question
```

Programme 4: Chainage avant – Parcours en largeur

On remarquera l'importance des lignes 10 et 14 pour le parcours en largeur : l'ajout ligne 10 de `r` en fin de l'ensemble contraint `EC` et l'exécution ligne 14 de la règle en tête de `EC` permettent d'ordonner les règles afin que les premières ajoutées soient les premières utilisées. On utilise ainsi *une structure de file* (aussi appelée FIFO pour "*First In, First Out*").

3.1.2 En profondeur

En modifiant légèrement le programme, on passe facilement du parcours en largeur au parcours en largeur.

```
1  (defun chainageAvantProf () ; Moteur chaînage avant en profondeur
2    (let (EC regleCourante)
3      (loop ; on boucle
4        (if (valeur (assoc 'Propositions *faits*)) ; si le but est présent dans
   ↪  la base de faits avec une valeur non nulle
5            (progn
6              (funcall 'afficherPropositions) ; on affiche les propositions
7              (return nil)) ; et on arrête
8            (dolist (r *regles*) ; sinon on parcourt les règles dans la base de
   ↪  règles
9              (when (declenchable? r); si une règle est déclenchable
10               (push r EC) ; on l'ajoute à l'ensemble contraint EN TÊTE
11               (setq *regles* (remove r *regles* :test 'equal))))) ; on l'enlève
   ↪  de la base de règles
12       (if EC ; si on peut encore déclencher des règles
13           (progn
14             (setq regleCourante (pop EC)) ; on choisit la dernière obtenue
15             (ajouter (conclusion regleCourante))) ; on ajoute son résultat à la
   ↪  base de faits
16           (askQuestion))))) ; sinon on pose une question
```

Programme 5: Chainage avant – Parcours en profondeur

Ce qui fait la différence ici se trouve à la ligne 10 : on ajoute cette fois-ci `r` en tête de `EC`. On passe alors d'une structure de file à une structure de pile (aussi appelée LIFO pour "*Last In, First Out*".

3.2 Fonctions outils

Afin d'abstraire les raisonnements nous avons mis au point des fonctions outils. `conclusion()`, `declenchable?()` et `ajouter()` sont celles mises en place pour les règles.

```lisp
;;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 valeur
      (push triplet *faits*))) ; sinon on rajoute triplet à la base de faits
```

Programme 6: Fonctions outils pour les règles

`objet()`, `operateur()` et `valeur()` sont celles mises en place pour les faits.

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

(defun operateur (triplet)
  (cadr triplet))

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

Programme 7: Fonctions outils pour les faits

3.3 Poser une question : fonction `askQuestion()`

Une possibilité lorsque le système n'arrive plus à tirer de conclusions et de poser des questions à l'utilisateur pour apporter de nouveaux faits. C'est ici le rôle réalisé par la fonction `askQuestion()`.

```
15 (defun askQuestion ()
16   (let ((attribut (car (set-difference (listeAttRegles) (listeAttFaits)))
   ↪  valeur))
17     ; "attribut" est le premier élément de la différence entre :
18     ; - la liste des attributs dans la base de faits
19     ; - la liste des attributs dans la base de règles (prémisses)
20     ; C'est-à-dire un attribut dont la valeur est inconnue.
21     (if attribut
22       (if (numberp (car (AttValues attribut))) ; la valeur de celui-ci
   ↪  est-elle un nombre ?
23           (until
24             (AND
25               (not (format t "Spécifiez : ~S~&Votre choix (nombre) : "
   ↪  attribut))
26               (numberp (setq valeur (read))))) ; Redemande tant que son choix
   ↪  n'est pas valide
27           (until
28             (AND
29               ; liste les valeurs possibles de l'attribut et fait lire un choix
   ↪  à l'utilisateur
30               (not (format t "Spécifiez : ~S~&~S~%Votre choix : " attribut
   ↪  (delete-duplicates (AttValues attribut))))
31               (member (setq valeur (read)) (delete-duplicates (AttValues
   ↪  attribut))))))
32               ; Redemande tant que son choix n'est pas valide
33       (error "Sorry, something went wrong"))
34     (pushnew (list attribut 'EQ valeur) *faits*)))
35     ; ajouter l'attribut / valeur à la base de regles
36     ; TODO : Gérer le "EQ"
```

Programme 8: Fonction `askQuestion()` permettant de récupérer des informations

Nous utilisons d'autres fonctions pour réaliser certaines tâches.

```
1  (defun listeAttFaits ()
2    (loop for fait in *faits*
3      collect (objet fait)))
4
5  (defun listeAttRegles ()
6    (loop for regle in *regles*
7      append (loop for premisse in (car regle)
8        collect (objet premisse))))
9
10 (defun AttValues (attribut)
11   (loop for regle in *regles*
12     if (assoc attribut (car regle))
13     collect (valeur (assoc attribut (car regle)))))
```

Programme 9: Fonctions outils pour `askQuestion()`

`listAttFaits()` récupère la liste des attributs . `askQuestion()`. `askQuestion()`.

3.4 Afficher les propositions du système : fonction `afficherPropositions()`

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

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

Programme 10: 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.