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 besoin (traiter de l'information, découvrir des concepts informatiques...), 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 toutes les technologies existantes 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/`.

En tant qu'étudiants ingénieurs en informatique, nous avons aussi sollicité certaines de nos connaissances dans le domaine.

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'inférence*.

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'inférence. 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 deux types de fonctionnement pour les moteurs d'inférence : 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 attribut, qui est le nom du concept que l'on veut modéliser dans le fait ;
2. une valeur, qui permet de quantifier l'attribut ;
3. un opérateur, qui permet de préciser la valeur de l'attribut

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

```
(attribut 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 exemples d'attributs que nous utiliserons pour modéliser les faits :

Attributs	Signification
`UserStory`	scénario initial de l'utilisateur
`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
`Paradigme`	précise le paradigme des bases de données

TABLE 1 – Exemple d'attributs pour les faits

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

2.3 Base de règles

Nous avons décidé d'implémenter les règles dans notre base de cette façon :

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

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

```
(defparameter *regles* '(

  ; Analyse du besoin

  (((UserStory EQ ResoudreProbMath) (Methode EQ Numerique))
    ((Application EQ Calcul-Numerique)))

  (((UserStory EQ ResoudreProbMath) (Methode EQ Formelle))
    ((Application EQ Calcul-Formel)))

  (((UserStory EQ Modeliser) (SystemeComplexe EQ Oui))
    ((UserStoryPrec EQ ModeliserSystemeComplexe)))

  (((UserStory EQ Modeliser) (SystemeComplexe EQ Non) (ModeliserDonnee EQ Oui))
    ((UserStoryPrec EQ ModeliserDonnee)))

    (((UserStoryPrec EQ ModeliserDonnee) (IntelligenceArtificielle EQ Symbolique))
      ((Application EQ Systeme-Expert)))

    (((UserStoryPrec EQ ModeliserDonnee) (IntelligenceArtificielle EQ Numerique))
      ((Application EQ Machine-Learning)))

    (((UserStoryPrec EQ ModeliserSystemeComplexe) (PrecisionSystemeComplexe EQ Equations))
      ((Application EQ Calcul-Numerique)))

    (((UserStoryPrec EQ ModeliserSystemeComplexe) (PrecisionSystemeComplexe EQ InteractionSys))
      ((Application EQ SMA)))

  (((UserStory EQ Decouvrir) (AimeJeu EQ Oui))
    ((UserStoryPrec EQ Jeux)))

    (((UserStoryPrec EQ Jeux) (Prefere EQ Smartphone))
      ((Application EQ Mobile)))

    (((UserStoryPrec EQ Jeux) (Prefere EQ PCetConsoles))
      ((Application EQ Jeu-video)))

  (((UserStory EQ Decouvrir) (AimeJeu EQ Non) (Internet EQ Non) (ParentheseDec EQ Non) (Programmation EQ Oui))
    ((Application EQ Apprentissage)))

  (((UserStory EQ Decouvrir) (AimeJeu EQ Non) (Internet EQ Oui) (Programmation EQ Non) (ParentheseDec EQ Non))
    ((Application EQ Site-Web)))

```

```
44   (((UserStory EQ Decouvrir) (AimeJeu EQ Non) (Internet EQ Non) (ParentheseDec
     ↪ EQ Oui))
45     ((Propositions EQ (LISP))))
46
47   (((UserStory EQ MiseEnFormeInfos) (BeaucoupInfo EQ NON))
48     ((Application EQ Redaction)))
49
50   (((UserStory EQ MiseEnFormeInfos) (BeaucoupInfo EQ OUI))
51     ((UserStoryPrec EQ TraitementDInfos)))
52
53   (((UserStoryPrec EQ TraitementDInfos) (PrecisionInfo EQ FiltrerTexte))
54     ((Application EQ Expression-Reguliere)))
55
56   (((UserStoryPrec EQ TraitementDInfos) (PrecisionInfo EQ Organiser))
57     ((Application EQ Dataware)))
58
59   (((UserStoryPrec EQ TraitementDInfos) (PrecisionInfo EQ FaireDesModeles))
60     ((Application EQ Machine-Learning)))
61
62   (((UserStory EQ DejaIdeeDev) (ChoixProjet EQ Site-web))
63     ((Application EQ Site-web)))
64
65   (((UserStory EQ DejaIdeeDev) (ChoixProjet EQ Logiciel))
66     ((Application EQ Logiciel)))
67
68   (((UserStory EQ DejaIdeeDev) (ChoixProjet EQ Applet))
69     ((Application EQ Applet)))
70
71   (((UserStory EQ DejaIdeeDev) (ChoixProjet EQ Jeu-video))
72     ((Application EQ Jeu-video)))
73
74   (((UserStory EQ DejaIdeeDev) (ChoixProjet EQ Machine-Learning))
75     ((Application EQ Machine-Learning)))
76
77   ; Site web
78   (((Application EQ Site-Web) (PrecisionSite EQ Simple) (Interaction-Dynamique
     ↪ EQ OUI))
79     ((Propositions EQ (HTML PHP MySQL Symfony JavaScript AJAX))))
80
81   (((Application EQ Site-Web) (PrecisionSite EQ Simple) (Interaction-Dynamique
     ↪ EQ NON))
82     ((Propositions EQ (HTML PHP MySQL))))
83
84   (((Application EQ Site-Web) (PrecisionSite EQ Responsive))
85     ((Propositions EQ (HTML PHP MySQL BootStrap JavaScript AJAX))))
86
87   (((Application EQ Site-Web) (PrecisionSite EQ Bonne))
88     ((Propositions EQ (HTML PHP MySQL Symfony))))
```

```
89
90    (((Application EQ Site-Web) (PrecisionSite EQ Efficace) (RoRvsDjango EQ
      ↪ Configurable))
91      ((Propositions EQ (HTML JavaScript Django AJAX))))
92
93    (((Application EQ Site-Web) (PrecisionSite EQ Efficace) (RoRvsDjango EQ
      ↪ Populaire))
94      ((Propositions EQ (HTML JavaScript Ruby Ruby-on-Rails AJAX))))
95
96    ; Application Mobile
97    (((Application EQ Mobile)(Machine EQ Mac)(Cible EQ iPhone) (Budget > 99))
98      ((Propositions EQ (Swift))))
99    (((Application EQ Mobile)(Machine EQ Mac)(Cible EQ iPhone) (Budget < 100))
100     ((Propositions EQ (Impossible))))
101
102   (((Application EQ Mobile)(Machine EQ Linux)(Cible EQ iPhone))
103     ((Propositions EQ (Impossible))))
104   (((Application EQ Mobile)(Machine EQ Windows)(Cible EQ iPhone))
105     ((Propositions EQ (Impossible))))
106
107   (((Application EQ Mobile) (Cible EQ Android))
108     ((Propositions EQ (JAVA Android-Studio))))
109
110   ; Logiciel
111   (((Application EQ Logiciel) (PrecisionLogiciel EQ Complexe))
112     ((Propositions EQ (C++ JAVA Git))))
113
114   (((Application EQ Logiciel) (PrecisionLogiciel EQ Complexe) (Machine EQ
      ↪ Windows))
115     ((Propositions EQ (C# Git))))
116
117   ; Calcul-Numerique
118   (((Application EQ Calcul-Numerique) (Budget > 100))
119     ((Propositions EQ (Matlab))))
120
121   (((Application EQ Calcul-Numerique) (Budget < 120) (ManipulationMatrice EQ
      ↪ OUI))
122     ((Propositions EQ (Octave Scilab Julia Fortran))))
123
124   (((Application EQ Calcul-Numerique) (Budget < 120) (ManipulationMatrice EQ
      ↪ NON))
125     ((Propositions EQ (Python MathPlotLib Numpy))))
126
```

```
127   ; Calcul-Numerique
128   (((Application EQ Calcul-Formel) (Budget < 100))
129     ((Propositions EQ (Sage))))
130
131   (((Application EQ Calcul-Formel) (Budget > 90))
132     ((Propositions EQ (Maple))))
133
134   ; Machine-Learning
135   (((Application EQ Machine-Learning) (PrecisionML EQ Prototypage-Rapide)
      ↪  (Budget < 101) (ManipulationMatrice EQ OUI))
136     ((Propositions EQ (Octave))))
137
138   (((Application EQ Machine-Learning) (PrecisionML EQ Prototypage-Rapide)
      ↪  (Budget < 101) (ManipulationMatrice EQ NON))
139     ((Propositions EQ (Python Sci-kit MathPlotLib Numpy))))
140
141   (((Application EQ Machine-Learning) (PrecisionML EQ Prototypage-Rapide)
      ↪  (Budget > 100))
142     ((Propositions EQ (Matlab))))
143
144   (((Application EQ Machine-Learning) (PrecisionML EQ Modele-Complexe) (Budget
      ↪  < 101))
145     ((Propositions EQ (R))))
146
147   (((Application EQ Machine-Learning) (PrecisionML EQ Modele-Complexe) (Budget
      ↪  > 100))
148     ((Propositions EQ (Matlab))))
149
150   ; Système-expert
151   (((Application EQ Systeme-Expert) (Parenthese EQ Oui))
152     ((Propositions EQ (LISP))))
153
154   (((Application EQ Systeme-Expert) (Parenthese EQ Non))
155     ((Propositions EQ (Prolog))))
156
157   ; Applet
158   (((Application EQ Applet) (Machine EQ Mac))
159     ((Propositions EQ (Swift))))
160
161   (((Application EQ Applet) (Machine EQ Linux))
162     ((Propositions EQ (Pygame Tkinter))))
163
164   (((Application EQ Applet) (Machine EQ Windows))
165     ((Propositions EQ (Pygame Tkinter))))
166
167
168   ; Jeu-Video
169   (((Application EQ Jeu-Video) (PrecisionJeu EQ 3D) (Budget < 35))
170     ((Propositions EQ (Impossible))))
171
```

```
172   (((Application EQ Jeu-Video) (PrecisionJeu EQ 3D) (Budget > 34))
173     ((Propositions EQ (C++ Unity3D OpenGL))))
174
175   (((Application EQ Jeu-Video) (PrecisionJeu EQ RPG-2D))
176     ((Propositions EQ (RPG-Maker))))
177
178   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Mac) (Budget <
      ↪ 35))
179     ((Propositions EQ (SpriteKit Swift Pygame))))
180
181   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Mac) (Budget >
      ↪ 34))
182     ((Propositions EQ (C++ Unity3D OpenGL))))
183
184   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Windows)
      ↪ (Budget < 35))
185     ((Propositions EQ (Pygame))))
186
187   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Windows)
      ↪ (Budget > 34))
188     ((Propositions EQ (C++ Unity3D OpenGL))))
189
190   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Linux) (Budget
      ↪ < 35))
191     ((Propositions EQ (Pygame))))
192
193   (((Application EQ Jeu-Video) (PrecisionJeu EQ 2D) (CibleJV EQ Linux) (Budget
      ↪ > 34))
194     ((Propositions EQ (C++ Unity3D OpenGL))))
195
196   ; DIY
197   (((Application EQ DIY) (AccesAInternet EQ OUI) (Capteur EQ OUI))
198     ((ObjetConnecte EQ OUI)))
199
200   (((Application EQ DIY) (ObjetConnecte EQ OUI) (CommunicationAvecAutres EQ
      ↪ OUI))
201     ((Propositions EQ (Raspberry-Pi))))
202
203   (((Application EQ DIY) (ObjetConnecte EQ OUI) (CommunicationAvecAutres EQ
      ↪ NON))
204     ((Propositions EQ (Arduino EthernetShield))))
205
206   (((Application EQ DIY) (AccesAInternet EQ NON) (CommunicationAvecAutres EQ
      ↪ OUI))
207     ((Propositions EQ (Arduino Bluetooth RadioTransmitter))))
208
209   (((Application EQ DIY) (AccesAInternet EQ NON) (CommunicationAvecAutres EQ
      ↪ NON))
210     ((ObjetConnecte EQ NON)))
211
```

```
212   (((Application EQ DIY) (ObjetConnecte EQ NON))
213     ((Propositions EQ (Arduino))))
214
215   ;Programmation systeme
216   (((Application EQ Systeme) (Machine EQ Linux) (PrecisionSysteme EQ
      ↪  Interaction))
217     ((Propositions EQ (C Shell Tkinter))))
218
219   ;Système Embarque
220   (((Application EQ Systeme-Embarque))
221     ((Propositions EQ (Assembleur Shell C))))
222
223   ;Système Multi-agents
224   (((Application EQ SMA) (PrecisionSMA EQ SimulationDeFoule))
225     ((Propositions EQ (MASSIVE))))
226
227   (((Application EQ SMA) (PrecisionSMA EQ Trading))
228     ((Propositions EQ (MetaTrader4))))
229
230   (((Application EQ SMA) (PrecisionSMA EQ Autre))
231     ((Propositions EQ (JADE Java))))
232
233   ;Dataware
234   (((Application EQ Dataware) (QuantiteDonnee > 100000))
235     ((Paradigme EQ NoSQL)))
236
237   (((Application EQ Dataware) (QuantiteDonnee < 100001))
238     ((Paradigme EQ Relationnel)))
239
240
241   (((Application EQ Dataware) (Paradigme EQ Relationnel) (AccesLecture > 1000)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ OUI))
242     ((Propositions EQ (HTML Django Rest PLSQL))))
243
244   (((Application EQ Dataware) (Paradigme EQ Relationnel) (AccesLecture < 1001)
      ↪  (Usage EQ Prive) (ControleAcces EQ OUI))
245     ((Propositions EQ (HTML Django PostGreSQL))))
246
247   (((Application EQ Dataware) (Paradigme EQ Relationnel) (AccesLecture > 1000)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ NON))
248     ((Propositions EQ (HTML Django Rest PLSQL OAuth2))))
249
250   (((Application EQ Dataware) (Paradigme EQ Relationnel) (AccesLecture < 1001)
      ↪  (Usage EQ Prive) (ControleAcces EQ NON))
251     ((Propositions EQ (HTML Django PostGreSQL OAuth2))))
252
253   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Document)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ OUI))
254     ((Propositions (HTML MongoDB Json Django Rest OAuth2))))
```

```
255
256   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Graphe)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ OUI))
257     ((Propositions (HTML Neo4J Django Rest OAuth2))))
258
259   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Document)
      ↪  (Usage EQ Prive) (ControleAcces EQ OUI))
260     ((Propositions (HTML MongoDB Json Rest OAuth2))))
261
262   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Graphe)
      ↪  (Usage EQ Prive) (ControleAcces EQ OUI))
263     ((Propositions (HTML Neo4J Rest OAuth2))))
264
265   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Document)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ NON))
266     ((Propositions (HTML MongoDB Json Django Rest))))
267
268   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Graphe)
      ↪  (Usage EQ GrandPublic) (ControleAcces EQ NON))
269     ((Propositions (HTML Neo4J Django Rest))))
270
271   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Document)
      ↪  (Usage EQ Prive) (ControleAcces EQ NON))
272     ((Propositions (HTML MongoDB Json Rest))))
273
274   (((Application EQ Dataware) (Paradigme EQ NoSQL) (Orientation EQ Graphe)
      ↪  (Usage EQ Prive) (ControleAcces EQ NON))
275     ((Propositions (HTML Neo4J Rest))))
276
277
278   ;RegExp
279   (((Application EQ Expression-Reguliere))
280     ((Propositions EQ (Perl JavaScript))))
281
282
283   ; Redaction
284   (((Application EQ Redaction) (PrecisionRedaction EQ Prototypage-Rapide)
      ↪  (UsageRedac EQ Collaboratif) (LogicielLibre EQ OUI))
285     ((Propositions EQ (Git MD Etherpad))))
286
287   (((Application EQ Redaction) (PrecisionRedaction EQ Prototypage-Rapide)
      ↪  (UsageRedac EQ Collaboratif) (LogicielLibre EQ PasForcement))
288     ((Propositions EQ (GoogleDoc))))
289
290   (((Application EQ Redaction) (PrecisionRedaction EQ Prototypage-Rapide)
      ↪  (UsageRedac EQ Individuel) (LogicielLibre EQ OUI))
291     ((Propositions EQ (MD Etherpad))))
292
293   (((Application EQ Redaction) (PrecisionRedaction EQ Prototypage-Rapide)
      ↪  (UsageRedac EQ Individuel) (LogicielLibre EQ PasForcement))
294     ((Propositions EQ (Word))))
```

```
295
296   (((Application EQ Redaction) (PrecisionRedaction EQ Complexe) (UsageRedac EQ
      ↪  Individuel) (LogicielLibre EQ PasForcement))
297     ((Propositions EQ (GoogleDoc Word))))
298
299   (((Application EQ Redaction) (PrecisionRedaction EQ Complexe) (UsageRedac EQ
      ↪  Individuel) (LogicielLibre EQ OUI))
300     ((Propositions EQ (LaTeX))))
301
302   (((Application EQ Redaction) (PrecisionRedaction EQ Complexe) (UsageRedac EQ
      ↪  Collaboratif))
303     ((Propositions EQ (Git LaTeX ShareLatex Overleaf))))
304
305   (((Application EQ Redaction) (PrecisionRedaction EQ Scientifique)
      ↪  (UsageRedac EQ Individuel))
306     ((Propositions EQ (LaTeX))))
307
308   (((Application EQ Redaction) (PrecisionRedaction EQ Scientifique)
      ↪  (UsageRedac EQ Collaboratif))
309     ((Propositions EQ (Git LaTeX))))
310
311   ; Apprentissage
312   (((Application EQ Apprentissage) (Recent EQ OUI))
313     ((Propositions EQ (Python Go))))
314
315   (((Application EQ Apprentissage) (Recent EQ PasForcement))
316     ((Propositions EQ (Perl Pascal))))
317 ))
```

Programme 1: Base de règles `*regles*`

2.4 Bases de connaissances

Nous avons construit 3 bases de connaissances pour notre système : une première pour les attributs, une seconde pour les questions, et une troisième pour les technologies. Celles-ci contiennent, pour chaque élément, une brève description de celui-ci.

Les éléments des ces bases sont représentés sous la forme de *liste pointée* ainsi :

```
(élément . "La description de l'élément")
```

Voici la première base de connaissances sur les attributs `*valeurs*` pour les valeurs des attributs que l'on peut choisir dans les questions : cela permettra à l'utilisateur d'avoir une idée plus précise sur les possibilités de réponses proposées.

```
(defparameter *Valeurs*
  '(
    (OUI . "Oui.")
    (NON . "Non.")
    (RESOUDREPROBMATH . "Je souhaite résoudre un problème mathématique.")
    (MODELISER . "Je souhaite modéliser un problème.")
    (MODELISERSYSTEMECOMPLEXE . "Je souhaite modéliser un système complexe.")
    (JEUX . "Oui j'aime les jeux")
    (DECOUVRIR . "Je souhaite découvrir des choses.")
    (MISEENFORMEINFOS . "Je souhaite mettre en forme de l'information.")
    (TRAITEMENTDINFOS . "Je souhaite traiter de l'information.")
    (APPRENTISSAGE . "Je veux apprendre un langage de programmation.")
    (SCIENTIFIQUE . "Le travail a un aspect scientifique et/ou académique.")
    (PASFORCEMENT . "Hmmm ... pas forcément.")
    (COLLABORATIF . "Je souhaite que celui-ci soit collaboratif.")
    (REDACTION . "C'est un travail de rédaction.")
    (EXPRESSION-REGULIERE . "Je souhaite utiliser des expression régulière.")
    (GRAPHE . "Je souhaite représenter les données sous forme de graphe.")
    (DOCUMENT . "Je souhaite représenter les données sous forme de document.")
    (NOSQL . "L'usage de la BDD est tournée vers le NoSQL.")
    (PRIVE . "Je souhaite en faire un usage privé.")
    (GRANDPUBLIC . "Je souhaite en faire un usage grand-public.")
    (RELATIONNEL . "L'usage de la BDD est tournée vers le NoSQL.")
    (DATAWARE . "Je souhaite traiter de vaste-ensemble de données.")
    (AUTRE . "Non, je veux faire autre choses avec les SMA.")
    (TRADING . "Je veux faire du trading, et/ou utiliser des technologies
    ↪ adaptées à la finance.")
    (SIMULATIONDEFOULE . "Je voudrais simuler le comportement de foule lors
    ↪ d'un évènement particulier.")
    (SMA . "Je veux utiliser des SMA.")
    (SYSTEME-EMBARQUE . "Je veux programmer un système embarqué.")
    (INTERACTION . "Je souhaite créer des fenêtres d'interactions entre
    ↪ l'utilisateur et la machine.")
    (SYSTEME . "Je souhaite définir des processus dans le système et/ou
    ↪ parametrer au mieux ma machine, planifier des tâches ...")
    (DIY . "Je veux bricoler un truc electronique.")
```

```
33      (LINUX . "Cela a pour visée Linux.")
34      (2D . "Mon jeu sera un jeu 2D.")
35      (RPG-2D . "Mon jeu sera un RPG-2D.")
36      (3D . "Le joueur évoluera dans un environnement 3D.")
37      (JEU-VIDEO . "Je veux développer un jeu-vidéo !")
38      (APPLET . "Je veux faire une petite application.")
39      (NON-SUPPORTEE . "Non ... je ne supporte pas trop les parenthèses.")
40      (SUPPORTEE . "Oui, cela ne me pose pas trop de problème.")
41      (SYSTEME-EXPERT . "Je veux faire un système-expert.")
42      (MODELE-COMPLEXE . "Je veux modéliser un phénomène ou une situation
    ↪ complexe.")
43      (PROTOTYPAGE-RAPIDE . "Je veux rapidement avoir quelque chose de
    ↪ fonctionnel et/ou d'expérimental et/ou facilement modifiable.")
44      (MACHINE-LEARNING . "Je préfère des méthodes d'apprentissage
    ↪ statistiques.")
45      (CALCUL-FORMEL . "Je veux résoudre ce problème de manière formelle.")
46      (INDIVIDUEL . "J'en ferrai une utilisation personnelle.")
47      (CALCUL-NUMERIQUE . "Je veux résoudre ce problème par des méthode
    ↪ numériques.")
48      (NUMERIQUE . "Je veux résoudre ce problème par des méthode numériques.")
49      (FORMELLE . "Je veux résoudre ce problème formellement.")
50      (WINDOWS . "Cela a pour visée un environnement Windows.")
51      (COMPLEXE . "Le document est un travail important.")
52      (LOGICIEL . "Je veux développer un logiciel.")
53      (ANDROID . "Je veux développer des app Android.")
54      (IPHONE . "Je veux développer des app iPhone.")
55      (MAC . "Je travaille sur un Mac.")
56      (MOBILE . "Je veux développer des applications smartphone.")
57      (POPULAIRE . "Je préfère une solution populaire chez les développeur avec
    ↪ une grande communauté.")
58      (CONFIGURABLE . "Je préfère une solution configurable où l'on peut
    ↪ comprendre et modifier à peu près tout.")
59      (EFFICACE . "Je veux faire site web d'une taille concéquante avec beaucoup
    ↪ d'utilisateurs l'utilisant.")
60      (RESPONSIVE . "Je veux faire un site web qui s'adapte bien à tout types
    ↪ d'appareil.")
61      (SIMPLE . "Je veux un site web simple et fonctionnel ; pas besoin qu'il
    ↪ soit spécialement beau.")
62      (BONNE . "Je veux un site web simple à mettre en place et d'assez bonne
    ↪ facture.")
63      (SITE-WEB . "Je veux faire un site-web.")
64      (FAIREDESMODELES . "Je veux faire des modèles prédictifs et/ou
    ↪ d'apprentissage sur ces données")
65      (ORGANISER . "Je souhaite organiser cette information.")
66      (FILTRERTEXTE . "Je veux chercher des informations spécifiques dans un
    ↪ ensemble de données textuelles éventuellement grand.")
67      (PCETCONSOLES . "Je préfère les jeux PC et consoles.")
68      (SMARTPHONE . "Je préfère les jeux sur smartphone.")
69      (EQUATIONS . "Je veux résoudre des équations.")
```

```
70      (SYMBOLIQUE . "Je préfère utilisée une intelligence artificielle
    ↪  symbolique.")
71      (INTERACTIONSYS . "Je veux simuler les interractions qu'il peut y avoir
    ↪  dans un système complexe.")
72      (DEJAIDEEDEV . "J'ai déjà une idée d'un projet à développer.")))
```

Programme 2: Base de connaissances *valeurs*

De même, nous avons défini une base de connaissances *questions* pour les questions à poser :cela permettra d'avoir une question adaptée pour chaque attribut lorsque l'on veut connaître la valeur du fait associé.

```
1  (defparameter *questions*
2  '(
3    (UserStory . "Que souhaitez-vous faire ? ")
4    (Application . "Quel type d'applications voulez-vous developper ou
    ↪  solliciter ?")
5    (PrecisionSite . "Précisez l'usage que vous voulez faire de votre site :")
6    (PrecisionSMA . "Précisez l'usage que vous voulez faire de votre système
    ↪  multi-agent :")
7    (PrecisionSysteme . "Précisez ce que vous souhaitez faire en programmation
    ↪  système :")
8    (PrecisionMachineLearning . "Précisez ce que vous voulez faire en machine
    ↪  learning :")
9    (PrecisionLogiciel . "Précisez l'usage que vous voulez faire de votre
    ↪  application :")
10   (PrecisionJeu . "Précisez l'usage que vous voulez faire de votre application
    ↪  :")
11   (PrecisionRedaction . "Dans quel cadre s'inscrit cette rédaction ?")
12   (PrecisionML . "Que souhaitez-vous faire ?")
13   (Machine . "Sur quel système d'explotation travaillez-vous ?")
14   (Cible . "Pour quel système developpez-vous ?")
15   (CibleJV . "Pour quel système developpez-vous ?")
16   (Budget . "Quel budget avez-vous ? Entrez un chiffre.")
17   (Usage . "De quel usage avez-vous besoin ?")
18   (UsageApp . "De quel usage avez-vous besoin pour votre applet ?")
19   (UsageRedac . "De quel usage avez-vous besoin pour la rédaction ?")
20   (ParentheseDec . "Voulez-vous découvrir un langage symbolique fondateur ?")
21   (Parenthese . "Voulez-vous découvrir un langage symbolique fondateur ?")
22   (ModeliserDonnee . "Voulez-vous modéliser un problème basée sur des données
    ↪  ?")
23   (QuantiteDonnee . "Votre projet nécessite-t-il de traiter beaucoup de
    ↪  données ?")
```

```
24    (Visiteurs . "Précisez le nombre de visiteurs attendus :")
25    (LogicielLibre . "Êtes-vous un fervant défenseur du logicel libre ?")
26    (Orientation . "Préfèreriez-vous que votre base de données soit orientée
      ↪  selon :")
27    (AccesLecture . "Combien d'accès en lecture se feront sur vos données ?
      ↪  Donnez le nombre d'accès aux données par heure.")
28    (Paradigme . "De quel paradigme de stockage de données s'agit-il ?")
29    (AccesAInternet . "Votre système aura-t-il besoin d'avoir accès à internet
      ↪  ?")
30    (ManipulationMatrice . "Aurez-vous besoin de manipuler des matrices ?")
31    (RoRvsDjango . "Préferez-vous une solution populaire ou configurable ?")
32    (Interaction-Dynamique ."Le design du site devra-t-il être dynamique ?")
33    (Internet . "Souhaitez-vous en apprendre plus sur Internet et sur comment
      ↪  faire un site-web ?")
34    (AimeJeu . "Souhaitez-vous développer un jeu ?")
35    (Recent . "Avez-vous une préférences pour des langages récents ?")
36    (Methode . "Comment souhaitez-vous résoudre ce problème ?")
37    (Programmation . "Voulez-vous apprendre à programmer ?")
38    (SystemeComplexe . "Voulez-vous modéliser un système complexe ?")
39    (PrecisionSystemeComplexe . "De quel type de systeme complexes s'agit-il ?")
40    (BeaucoupInfo . "Disposez-vous de beaucoup d'informations ? ")
41    (Prefere . "Que préférez-vous entre les jeux PC et les jeux smartphone ?")
42    (ControleAcces . "Avez-vous besoin de contrôler l'accès de différents
      ↪  utilisateurs sur cette plateforme ?")
43    (ChoixProjet . "Quel est votre projet ?")
44    (IntelligenceArtificielle . "Quel paradigme d'IA souhaitez-vous utilisez
      ↪  ?")))
```

Programme 3: Base de connaissances `*questions*`

Enfin, voici la base de connaissances sur les 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.

```
(defparameter *technologies*
'(
  ; Programmation système
  (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 et pour de nombreuses autres choses en programmation système.")
  (Shell . "Interpréteur de commandes : l'interface entre l'OS et l'utilisateur ; il existe différente famille, le plus utilisé restant 'bash'.")
  (Assembleur . "L'assembleur est un des langage de plus bas niveau qui convertit des instructions en langages machine.")


  ; Python, framework & bibliothèque
  (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 . "Bibliothèque python : boîte à outils scientifiques.")
    (MathPlotLib . "Bibliothèque python : utile pour la visualisation de données tracé de courbes, de surfaces, de nuages de points....")
    (Sci-kit . "Bibliothèque python : dispose de beaucoup d'implémentation d'algorithme de machine learning.")

  ; Calcul Numérique & Machine learning
  (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.")
  (Scilab . "Logiciel et langage de programmation de calcul numérique, alternative libre et gratuite à Matlab.")
  (Julia . "Logiciel et langage de programmation de calcul numérique, alternative récente et libre à Matlab. Performant.")

  ; Calcul formel
  (Maple . "Logiciel de calcul formel ; propriétaire et payant.")
  (Sage . "Se veut être ''une alternative viable libre et open source à Magma, Maple, Mathematica et Matlab''. C'est un langage de calcul formel.")

```

```
29   ; Intelligence artificielle & ontologies
30   (LISP . "Le langage de programmation pour intellience artificielle
     ↪ symbolique le plus populaire de tous. Il est minimaliste et permet
     ↪ d'implémenter de nombreuses choses.")
31   (Prolog . "Un langage de programmation français (Cocorico !), il est utilisé
     ↪ en Intelligence Artificielle.")
32   (OWL-Lite . "Langage de représentation des connaissances construit sur le
     ↪ modèle de données de RDF. Basé sur la logique de description, Web Ontology
     ↪ Language (OWL) permet de définir des ontologies web structurées. OWL-Lite
     ↪ en est sa version la plus simple, pour les utilisateurs ayant besoin d'une
     ↪ hiérarchie de classification et de contraintes simples.")
33   (OWL-DL . "Langage de représentation des connaissances construit sur le
     ↪ modèle de données de RDF. Basé sur la logique de description, Web Ontology
     ↪ Language (OWL) permet de définir des ontologies web structurées. OWL-DL en
     ↪ est une version intermédiaire décidable qui permet une expressivité
     ↪ maximale avec une garantie de calcul.")
34   (OWL-Full . "Langage de représentation des connaissances construit sur le
     ↪ modèle de données de RDF. Basé sur la logique de description, Web Ontology
     ↪ Language (OWL) permet de définir des ontologies web structurées. OWL-Full
     ↪ en est sa version la plus libre, indécidable, permettant une expressivité
     ↪ maximale mais sans garantie de calcul.")
35
36   ; Systèmes multi-agents
37   (MASSIVE . "Logiciel utilisant les Systèmes Multi-agents (SMA) dans le but
     ↪ de simuler des foules. Il est très utilisé dans les industries du cinéma
     ↪ et du jeu vidéo, notamment dans la trilogie Seigneur des Anneaux ainsi que
     ↪ dans la série Games of Thrones.")
38   (MetaTrader4 . "Logiciel utilisant les Systèmes Multi-agents (SMA) dans le
     ↪ domaine de la finance afin d'effectuer du trading automatique.")
39   (JADE . "JAVA Agent DEvelopment (JADE) est une plateforme de programmation
     ↪ multi-agent implémentée en Java.")
40
41   ; Bases de données relationnelles
42   (PLSQL . "Variante propriétaire d'Oracle ; permet d'implémenter des bases de
     ↪ données relationnelles-objet et dispose de fonctions et fonctionnalités
     ↪ supplémentaires.
43    Les bases de données relationnelles-objet sont très efficaces lorsqu'il
     ↪ s'agit de construire des bases de données dont l'accès en lecture est très
     ↪ fréquent.")
44   (PostGreSQL . "Variantes libre d'implémentation de SQL la plus populaire.
     ↪ Elle reste la plus plebiscitée car elle est très robuste en terme
     ↪ d'implémentation et veille à suivre de très près les standard SQL.")
45   (MySQL . "Variantes libre d'implémentation de SQL, simple d'utilisation mais
     ↪ aussi moins bien structurée.")
46
47   ; Bases de données NoSQL
48   (Neo4J . "Technologie NoSQL : données représentées sous forme de graphes.")
49   (MongoDB . "Technologie NoSQL : données non structurées stockées sous le
     ↪ formalisme JSON")
50
```

```
51   ; DIY
52   (Arduino . "Cartes électroniques de 'hacking' en license libre. Elles se
     ↪ programment généralement dans un formalisme proche du C et du C++")
53   (EthernetShield . "Module qui permet à un Arduino de ce connecter à
     ↪ Internet.")
54   (Bluetooth . "Protocole de communication qui peut être facilement utilisé
     ↪ pour des connexions avec d'autres objets via des modules.")
55   (RadioTransmitter . "Une solution simple pour communiquer entre des objets,
     ↪ moins onéreuse que le Bluetooth.")
56   (Raspberry-Pi . "Mini-ordinateur très utilisé pour réaliser des petits
     ↪ serveurs dans des projets DIY.")
57
58   ; Web
59   (HTML . "Formalisme de représentation de données utilisée par les pages web.
     ↪ La base du développement web depuis toujours. Il peut être aussi utilisé
     ↪ pour créer des documents.")
60     (JavaScript . "Langage de programmation permettant de faire fonctionner
     ↪ des applications web côté client.")
61       (AJAX . ", pour 'Asynchronous JAvascript and Xml'. Architecture qui
     ↪ permet de créer des applications dynamiques. Est utilisé pour les sites
     ↪ web dynamiques mais aussi pour les systèmes multi-agent")
62     (PHP . "Le langage de developpement web le plus utilisé.")
63       (Symfony . "Sûrement le framework PHP le plus populaire : formalisme
     ↪ Modèle-Vue-Contrôleur")
64     (Json . "Un formalisme récent de représentation simple et lisible
     ↪ d'informations ; standard utilisé par beaucoup de langages.")
65     (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")
66     (Ruby-on-Rails . "Le framework Ruby pour le developpement web. Plus
     ↪ populaire que Django.")
67     (BootStrap . "Des feuilles de style CSS utilisées par les développeurs qui
     ↪ ne veulent rapidement un site beau et Responsive")
68     (Rest . "Technologie très utilisée pour faire des API.")
69     (OAuth2 . "Protocole d'authentification utilisé pour les sites web.")
70
71   ; Programmation de logiciel et d'applications smartphone
72   (Swift . "Le dernier langage de programmation d'Apple pour développer des
     ↪ applications iPhone et Mac")
73   (C# . "Le langage de programmation orientée objet de Microsoft. Il révèle
     ↪ tout son potentiel s'il est utilisé conjointement au framework .NET.
     ↪ Intéressant uniquement si la développement se fait dans un envirronement
     ↪ Microsoft.")
74   (Java . "Le langage de programmation orienté objet")
75   (C++ . "Inspiré du C, il en reprend beaucoup de spécificités et est orienté
     ↪ programmation objet. C'est un langage très plebiscité pour la création de
     ↪ logiciel complexe.")
76   (Android-Studio . "Android Studio est l'une des principale plateforme de
     ↪ développement d'applications pour machine sous Android.")
77
```

```
78    ; Jeu-Video
79    (Unity3D . "Le moteur graphique le plus intéressant pour développer des jeux
      ↪  en 3D avec des graphiques époustouflants !")
80    (OpenGL . "Un moteur graphique Open-Source utilisé pour développer beaucoup
      ↪  de jeux.")
81    (RPG-Maker . "L'outil le mieux adapté pour réaliser des jeux RPG en 2D.")
82    (SpriteKit . "L'outil le mieux adapté pour développer des jeux en 2D sous
      ↪  Mac.")
83
84    ; Rédaction et versionnage
85    (LaTeX . "Langage et un système de composition de documents. Utilisé pour la
      ↪  rédaction de documents scientifique. Beaucoup de bibliothèque (package)")
86    (Git . "Logiciel de versionnage libre ; très puissant pour travailler sur
      ↪  des projets à plusieurs ; permet de minimiser les collisions entre
      ↪  versions.")
87    (Word . "Logiciel propriétaire WYSIWYG de Microsoft. Le plus populaire pour
      ↪  la rédaction du document")
88    (GoogleDoc . "Solution en ligne de Google. Permet des rédactions de
      ↪  documents à plusieurs très interactive. Nécessite d'être connecté à
      ↪  Internet.")
89    (MD . "Une extension de fichier (MD pour 'Markdown Documentation') surtout
      ↪  utilisée pour rédiger des documentations. Les fichiers MD sont facilement
      ↪  interprétable et leur syntaxe est simple d'usage")
90    (Etherpad . "Solution libre à GoogleDoc ; est perfectible mais fait le
      ↪  travail.")
91    (ShareLatex . "Un éditeur de document LaTeX gratuit disponible en ligne :
      ↪  très utile pour la rédaction de documents collaborative.")
92    (Overleaf . "Un autre éditeur de document LaTeX en ligne : il propose des
      ↪  templates intéressants.")
93
94    ; Autres langages de programmation
95    (Fortran . "Vieux langage de programmation utilisé pour le calcul
      ↪  scientifique")
96    (Go . "Langage de programmation de Google ; se veut efficace et simple
      ↪  d'apprentissage.")
97    (Perl . "Un langage de programmation simple à apprendre. Utile pour
      ↪  déterminer des expressions régulières")
98    (Pascal . "Un langage de programmation ancien; sa &xe est simple ce qui lui
      ↪  donne un bon atout pédagogique.")
99    )
100 )
```

Programme 4: Base de connaissances `*technologies*`

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'attributs 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 5: 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 6: 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
(defun conclusion (r)
  ; Renvoie les conclusions d'une règle
  (cadr r))

(defun declenchable? (r)
  ; Renvoie t si la règle est déclenchable ; nil sinon
  (let (
      (OK t)
      (premisses (car r)))
    (dolist (p premisses OK) ; pour chaque premisses de r
       ; si l'attribut a une valeur numérique mais n'est pas présent dans les faits
      (if (and (numberp (valeur p)) (not (valeur (assoc (attribut p) *faits*))))
        (setq OK nil)
         ; s'il est présent mais si sa valeur ne vérifie pas la prémisse p
        (if (not (funcall (operateur p) (valeur (assoc (attribut p) *faits*)) (valeur p)))
          (setq OK nil))))))

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

Programme 7: Fonctions outils pour les règles

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

```lisp
(defun attribut (triplet)
  (car triplet))

(defun operateur (triplet)
  (cadr triplet))

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

Programme 8: 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()`.

```
(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))) ; la valeur de celui-ci est-elle un nombre ?
          (until
            (AND
              (not (format t "------~&~S~%------~%Votre choix (nombre) : " (questionAssociee 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 "------~&~S~%------~%~S~%~%Votre choix : " (questionAssociee attribut) (afficherchoix (delete-duplicates (AttValues attribut)))))
              (member (setq valeur (read)) (delete-duplicates (AttValues attribut))))))
              ; Redemande tant que son choix n'est pas valide
      (error "Nous n'avons rien pu trouver."))
    (sleep 1)
    (pushnew (list attribut 'EQ valeur) *faits*)))
```

Programme 9: Fonction `askQuestion()` permettant de récupérer des informations

Nous utilisons d'autres fonctions pour réaliser certaines tâches ; les voici :

```
24 (defun listeAttFaits ()
25   ; retourne la liste des attributs
26   ; présents dans la base de faits
27   (loop for fait in *faits*
28     collect (attribut fait)))
29
30 (defun listeAttRegles ()
31   ; retourne la liste des attributs non valués dans les règles
32   ; ayant des attributs déjà valués
33   (if *faits*
34     ;pour chaque règle contenant un attribut valué, récupérer les attributs
35     (loop for regle in
36       ;constitution de la liste des règles contenant un attribut valué
37       (loop for fait in *faits*
38         append
39           (loop for regle in *regles*
40             if (member fait (car regle) :test 'equal)
41               ;si la règle contient un attribut valué, la collecter
42               collect regle))
43       append
44         ;Récupération des attributs de la règle
45         (loop for premisse in (car regle)
46           collect (attribut premisse)))
47     ;Si aucun fait n'est valué (présent dans *faits*), sélectionner tous les
   ↪ attributs
48     (loop for regle in *regles*
49       append (loop for premisse in (car regle)
50           collect (attribut premisse)))))
51
52 (defun AttValues (attribut)
53   ; retourne la liste des valeurs
54   ; qu'un attribut peut prendre
55   (loop for regle in *regles*
56     if (assoc attribut (car regle))
57     collect (valeur (assoc attribut (car regle)))))
58
59 (defun questionAssociee (attribut)
60   ; retourne la question associée
61   ; à un attribut dans la base *questions*
62   ; ou celui-ci si cette dernière n'est pas présente
63   (or (cdr (assoc attribut *questions*)) attribut))
64
65 (defun afficherChoix (listeAttribut)
66   ; renvoie une string pour apporter une précision
67   ; sur les différents choix
68   (setq stringRetour "")
69   (loop for a in listeAttribut
70     do (setq stringRetour (concatenate 'string stringRetour "~% ->
   ↪ "(symbol-name a) " : "(descriptionAttribut a))))
71     (format nil stringRetour))
```

Programme 10: Fonctions outils pour `askQuestion()`

```
(defun descriptionAttribut (attribut)
  ; retourne la description associée
  ; à un attribut dans la base *valeurs*
  ; ou celle-ci si cette dernière n'est pas présente
  (or (cdr (assoc attribut *valeurs*)) (symbol-name attribut)))
```

Programme 11: Fonctions outils pour `askQuestion()`

`listAttFaits()` récupère la liste des attributs sur présents dans la base de faits.

`listeAttRegles()` récupère quant à elle la liste des attributs sur lesquels il faut poser des questions.

`AttValues()` se charge de trouver les valeurs que les différents attributs peuvent prendre.

`questionAssociee()` retourne la question associée à un attribut. `afficherChoix()` retourne une string comportant les descriptions des différentes valeurs que l'utilisateur peut saisir ; elle utilise la fonction `descriptionAttribut()` qui retourne pour chaque valeur dans la base `*valeurs*` la description associée à celle-ci.

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

C'est dans cette fonction que nous utilisons la base de connaissances `*technologies*`. Elle signe la fin de l'exécution du système expert.

```
(defun afficherPropositions ()
  ; Affichage final des propositions
  (if (member 'Impossible (caddr (assoc 'Propositions *faits*)))
    (format t "~%Il n'y a pas de solution possible dans votre cas, désolé.")
  (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 (descriptionTechno x)))
    (format t "~&~A~&" overlay))))
```

Programme 12: Fonction `afficherPropositions()` qui affiche les propositions du système expert

Cette fonction utilise la fonction outils `descriptionTechno()`.

```
(defun descriptionTechno (techno)
  ; retourne la description associée
  ; à une technologie dans la base *technologies*
  ; ou celle-ci si cette dernière n'est pas présente
  (or (cdr (assoc techno *technologies*)) techno))
```

Programme 13: Fonction `descriptionTechno()` : retourne la description d'une technologie

On remarquera que le système affichera qu'il n'y a pas de solutions possibles lorsque `Impossible` est présent comme propositions dans la base de faits. En effet, il existe des situations qui ne peuvent pas être vérifiée dans la vie réelle ; par exemple, on ne peut pas développer des applications *iPhone* si l'on a pas de machine sous *MacOSX* (cf. règles présentes au lignes 102 et 103 dans la base de règles `*règles*`).

⋆ ⋆ ⋆

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.