Introduction aux Systèmes Experts
Version 1.0.3*

Rolland Balzon Philippe
Department of Computer Science
λ^∞ Free Corp.
prolland@free.fr

24 juillet 2002

Résumé

Il s'agit de rappeler les notions de bases des Systèmes Experts. Notemment on précise le domaine d'application, les éléments théoriques, ainsi que les languages utilisés pour implémenter un système expert.

Table des matières

*This document can be found to http ://prolland.free.fr/works/ai/docs/intro.pdf

1 Domaine d'application

On présente quelques caractéristiques des problèmes traités par les systèmes à base de connaissance :

- problèmes plus QUALITATIFS que quantitatifs,
- données souvent SYMBOLIQUES,
- données qui EVOLUENT, parfois INCERTAINES souvent INCOMPLETES,
- traitement ayant recours à une HEURISTIQUE

2 Eléments de logique des propositions

On distingue deux catégorie d'entrées pour un système expert : les proposition et les règles d'inférence.

2.0.1 Propositions

Par pédagogie on donne de suite un exemple de propositions :

- `P1` : "les bébés pleurent quand ils ont faim"
- `P2` : "il pleut"
- `P3` : "je prends mon parapluie"
- `P4` : "il neige"
- `P5` : "les roses sont bleues"
- `P6` : "s'il pleut, je prends mon parapluie"

Ces expressions sont considérées indépendamment du fait de savoir si elles sont vraies ou fausses. La validité d'une proposition P quelconque est donnée par sa "valeur de vérité" : `val(P)`. La valeur de vérité est donc une fonction de l'ensemble des propositions sur l'ensemble à deux éléments : "`vrai`" et "`faux`".

La valeur de vérité d'une proposition peut dépendre du lieu, de l'époque, etc. Par exemple `val(P5) = faux` (!). On peut aussi associer à chaque proposition une valeur numérique représentant un degré de certitude.

A partir de propositions, il est possible de construire d'autres propositions à l'aide de diverses opérations. En particulier :

- La conjonction (ET, $\wedge$). Exemple : `P2` $\wedge$ `P3` : "il pleut et je prends mon parapluie"
- La négation (NON, $\neg$) . Exemple : $\neg$`P4` : "il ne neige pas"
- La disjonction (OU, $\vee$). Exemple : `P2` $\vee$ `P4` : "il pleut ou il neige"
- L'implication (SI ... ALORS, $\Longrightarrow$) . Exemple : `P2` $\Longrightarrow$ `P3(= P6)`

Ainsi les propositions se combinent entre elles. C'est en ce sens que l'on parle d'"algèbre" des propositions.

Si `val(P)` est connue et si `val(Q)` est connue également, alors il n'est pas possible d'associer une valeur de vérité arbitraire à l'une de leur combinaison. Ce sont des "tables de vérité" qui indiquent, par exemple, la valeur de `val(P` $\wedge$ `Q)` connaissant `val(P)` et `val(Q)` (voir le cours de mathématique).

2.0.2 Règle d'inférence

Une autre notion introduite par la logique, plus importante encore, est celle de règle d'inférence (ou règle de déduction logique). On retiendra deux règles d'inférence :

- **Modus Ponens** : `Si val(P) = vrai et val(P`$\Longrightarrow$`Q) = vrai` on en déduit : `val(Q) = vrai`.
  C'est ce mode de raisonnement qui nous indique que si "il pleut" et que "chaque fois qu'il pleut, Monsieur Simon prend son parapluie", alors "Monsieur Simon doit être muni actuellement de son parapluie"
- **Modus Tollens** : `Si val(P`$\Longrightarrow$`Q) = vrai et val(Q) = faux` on en déduit : `val(P) = faux`.

Par la suite on introduira le vocable de FAIT pour les propositions de base et l'on dira qu'une implication `P` $\Longrightarrow$ `Q` est une REGLE. `P` est appelée la prémisse de la règle et `Q` la conclusion.

2.1 Examen d'un exemple

Voici les instructions (extrait) qui permettent de déterminer la nature d'une panne de démarreur (tirées de Volkswagen ServiceRepair Handbook, Clymer Publications, Los Angeles 1974).

"... Premier cas : le démarreur ne fonctionne pas ou fonctionne au ralenti. Dans ce cas, il est proposé d'allumer les grands phares. Si ceuxci sont faibles, c'est un problème de batterie. Si les phares fonctionnent normalement, on propose alors d'essayer de faire démarrer le moteur avec les phares allumés. Si la lumière baisse, il y a un courtcircuit dans le démarreur. Par contre, si la lumière des phares n'est pas modifiée, on propose de relier alors les deux grandes bobines. Si le moteur part, il y a un problème de bobine. Sinon la panne est autre.

Deuxième cas : le démarreur fonctionne, mais n'entraîne pas le moteur. C'est alors un problème de pignon..."

On trouve dans cette exemple des faits, par exemple :

- `F1` : "le démarreur ne fonctionne pas"
- `F2` : "les grands phares sont faibles"
- `F3` : "problème de batterie"

Une règle relie ces trois faits :

`R4 : (F1` $\wedge$ `F2)` $\Longrightarrow$ `F3`

C'est par le modus ponens que l'on déduit que `val(F3) = vrai` lorsque `val(F1` $\wedge$ `F2) = vrai`, `val(R4)` étant vrai ... selon le constructeur. Certaines parties du texte peuvent présenter des informations d'une autre nature; ce ne sont pas toujours des faits significatifs du problème considéré. Par exemple : "il est proposé d'allumer les grands phares" est une information complémentaire que l'on traitera à part.

La règle R4 se réécrira :

`SI le démarreur ne fonctionne pas ET grands phares faibles ALORS problème de batterie`

Exercice : Distinguer dans ce texte les autres faits, règles et informations complémentaires (attention, "si ... alors" ne donne pas toujours lieu à une règle ...).

2.2 Logique d'ordre 0 et logique d'ordre 1

Prenons le syllogisme classique suivant :

`P1` : "jean est un homme" `P2` : "les hommes sont mortels"

De ces deux propositions on en déduit `P3` : "jean est mortel".

Selon quel mécanisme peuton faire une telle déduction ?

Ces propositions font intervenir des classes (ou ensembles), celle des hommes et celle des créatures mortelles. De la proposition universelle `P2(X)` : "Si est X est un homme alors X est mortel", on en déduit la proposition `P3` à l'aide de `P1`, de `P2(jean)` et du Modus ponens. La logique utilisée introduit donc des variables.

Une logique sans variable est dite d'ordre 0. Une logique qui admet des variables est dite d'ordre 1. Les systèmes experts seront d'ordre 0 ou 1 selon qu'ils peuvent utiliser ou non des variables dans les propositions qu'ils manipulent.

2.3 Définition et catégorisation des systèmes experts

En reprenant la définition de Farreny [FARRENY], légèrement complétée, on dira qu'un système expert est une machine déductive relativement générale exploitant une collection séparée, sujette à évolution, d'unités de savoirfaire concernant un domaine particulier d'expertise humaine. Son but est d'apporter des solutions à des problèmes bien délimités concernant le domaine en question. Un tel programme assure aussi des fonctions complémentaires de dialogue, d'apprentissage et d'explication de son comportement.

Il y a différents types de systèmes experts (SE). E. G. Aillaud en a recensé cinq, qu'il a coté d'un coefficient de difficulté allant de 1 à 5. Un SE de coefficient de difficulté 1 est relativement facile à réaliser, alors qu'un système de niveau 5 est encore au stade de la recherche [AILLAUD].

- Automatisation de manuels, de procédures, de règlements, ... (1,3)
- Recherche d'informations pertinentes dans une grande quantité de données (domaine des SGBD déductives) (1,3)
- Aide à la décision, recherche opérationnelle, logistique (2,4)
- Recherche de solutions à des problèmes peu formalisés (langage naturel, diagnostics complexes) (1,5)
- Education, formation, apprentissage (1,5)

3 Organisation des systèmes à règles de production

Un système expert est constitué de trois "composants" principaux :

3.1 Base de faits

c'est l'ensemble des propositions connues du système à un moment donné. La base de faits est une mémoire de travail. Son contenu dépend du problème traité.

3.2 Base de connaissances

Il s'agit de la connaissance "experte" proprement dite. Dans les systèmes (simples) que nous considérons, cette base est constituée d'un ensemble de règles (P$\Longrightarrow$Q) qui mettent en relation des faits dans un domaine donné.

3.3 Moteur d'inférence

Moteur d'inférence : c'est le programme informatique qui combine les faits établis et les règles pour établir de nouveaux faits.

4 Moteur d'inférence

On a déjà vu qu'un système expert pouvait être d'ordre 0 ou d'ordre 1 (d'autres ordres 0+, 0++, 1+ ont été introduits pour rendre compte de dispositifs particuliers). De nombreuses autres caractéristiques différencient les systèmes existants. Une de cellesci est la façon d'utiliser les règles : de la prémisse vers la conclusion (chaînage avant) ou de la conclusion vers la prémisse (chaînage arrière).

A titre d'exemple considérons la base de connaissance constituée de trois règles :

REGLE r1	REGLE r2	REGLE r3
SI animal vole ET animal pond des oeufs ALORS animal est un oiseau	SI animal a des plumes ALORS animal est un oiseau	SI animal est un oiseau ET animal a de longues pattes ALORS animal est une autruche

Par ailleurs on suppose que la base de faits est constituée de :

F1	F2	F3
animal a des plumes	animal a un long cou	animal a de longues pattes

Les deux types de navigation d'un moteur d'inférence sont le chaînage avant et chaînage arrière.

4.1 Chaînage avant

En chaînage avant, le moteur sélectionnera et déclenchera les règles dont les prémisses sont satisfaites.

4.2 Chaînage arrière

En chaînage arrière, le système part sur un but à démontrer : par exemple F5 : "animal est une autruche". Il sélectionnera alors, de proche en proche les règles qui permettent de prouver ce fait.

4.3 Distinguer Chaînage avant et arrière

On peut décrire le fonctionnement d'un moteur en chaînage avant ou en chaînage arrière en considérant une seule règle, par exemple :

- Si la conclusion est ajoutée à la base de fait dès que la prémisse est satisfaite, c'est du chaînage avant.
- Si la conclusion n'est ajoutée à la base de fait que lorsqu'une requête est faite à son sujet, il s'agit alors de chaînage arrière.

5 Languages

5.1 Lisp

Le langage LISP a été conçu par John McCarthy entre les années 1956 et 1958. Une première implantation du langage (version 1.5) a été réalisée au MIT sur un ordinateur IBM 704 entre 1958 et 1962. Il existe plusieurs versions de LISP. Une référence est Common Lisp [STEELE] qui est compatible avec de nombreuses implantations et dont d'autres tentent de s'approcher : Le Lisp [CHAILLOUX]. Une version intéressante, du domaine public, XLISP. Une version orientée pour l'enseignement est SCHEME.

Les objets du langage sont principalement les atomes : `qwerty, a2` les nombres : `123.8`; les listes : `(3 et 4 donnent 7)`. Toutes les manipulations de données s'effectuent à l'aide de fonctions. Quelques exemples :

```
(plus 3 4) > 7
(car '(3 et 4 donnent 7)) > 3
(cdr '(3 et 4 donnent 7)) > (et 4 donnent 7)
```

Les opérateurs peuvent s'enchaîner : `(car (cdr '(3 et 4 donnent 7))) > et`

Le Lisp est caractérisé par l'équation Données = Programmes. En effet les programmes sont des listes. Il existe une fonction (eval) qui permet d'évaluer une liste en tant que programme. En Lisp, programmer consiste à définir de nouveaux opérateurs.

Exemple, la fonction fibo

La suite de fibonacci 1, 1, 2, 3, 5, 8, 13, 21, ... est donnée par : fibo(0) = 1, fibo(1)=1 et la relation : fibo(n)= fibo(n-1) + fibo(n-2). Sa définition en Lisp est :

```
(defun fibo (n)
(cond ((< n 2) 1)
(t (+ (fibo (- n 1))
(fibo (- n 2))))
)
)
```

Les systèmes Lisp comptent de quelques dizaines d'opérateurs primitifs à plus de mille.

5.2 Prolog

Les objets primitifs sont du même type qu'en Lisp, les atomes, les nombres et les listes ([3,et,4,donnent,7]).

Programmer c'est "déclarer" des assertions ou des faits, définir des règles et poser des questions.

- `précieux(or)`.
- `précieux(diamant)`.
- `substance(or,métal)`.
- `substance(diamant,minéral)`.

La liste précédentes fixent des faits bâtis sur les prédicats : précieux, substance, etc. Ils se lisent : l'or est précieux, l'or est un métal, etc. Ces prédicats correspondent aux tables des systèmes relationnels.

Il est possible d'effectuer des requêtes, par exemple :

- `?précieux(or)` $\Longrightarrow$ yes (la requête pour savoir si l'or est précieux a abouti)
- `?précieux(argent)` $\Longrightarrow$ no (elle a échoué)
- `?précieux(`X`)` $\Longrightarrow$ $\{X$=or, X=diamant$\}$ (la requête abouti et lie X aux valeurs possibles)

Les requêtes peuvent s'enchaîner :

`?précieux(`X`),substance(`X,Y`)` (de quelle substance (Y) sont les choses précieuses (X))

Il est possible de donner des règles, par exemple :

`même substance(X,Y) :- substance(X,Z),substance(Y,Z)`

qui se lit même substance(X,Y) si substance(X,Z) et substance(Y,Z) et qui signifie : X et Y sont liés par même substance si ils ont même substance (Z).

Fibo en Prolog

```
fibo(N,1):,N<2,!.
fibo(N,R):
N1 is N1,N2 is N2,
fibo(N1,R1),fibo(N2,R2),
R is R1+R2.
```

Il met en évidence la "coupure" : "!" et l'opérateur d'évaluation : "is".

5.3 Logo et μSIMP

Ils sont semblables à Lisp. Une liste en μ-SIMP s'écrit : (4.(et.(3.(font.7)))) (c'est la forme archaïque des paires de Lisp). En Logo on a : [4 et 3 font 7]

Le programme FIBO en μ-SIMP

```
FUNCTION FIBO(N)
WHEN N<2, 1 EXIT,
FIBO(N1)+FIBO(N2),
ENDFUN;
```

Le programme FIBO en LOGO

```
POUR FIBO :N
SI N<2 SORS 1
SORS SOMME FIBO :N1 FIBO :N2
FIN
```

5.4 SmallTalk

C'est le prototype des langages à objets. SMALLTALK est un environnement livré avec plusieurs centaines de classes prédéfinies et des outils (browser) qui permettent de l'enrichir, donc de développer des applications. Pour définir la fonction de Fibonacci en SMALLTALK, il faut sélectionner la classe Integer et ajouter une méthode du type :

```
fibonacci
^self < 2
ifTrue: [1]
ifFalse: [(self-1)fibonacci + (self-2)fibonacci]
```

Le calcul du dixième nombre de fibonacci se fera en invoquant sur un nombre la méthode de sélecteur fibonacci : 10 fibonacci. La méthode renvoie ($\hat{}$) le résultat de l'envoi du message fibonacci au nombre 10.

6 Glossaire

Assertion : proposition admise comme vraie. "Jean aime Marie" est une assertion. Plusieurs systèmes d' AI travaille à partie de telles assertions.

Backtracking (retour en arrière) : importante stratégie de contrôle dans les systèmes d' IA. Un backtrack intervient dans une recherche systématique de type combinatoir. Lorsque dans une condition donnée toutes les possibilités ont échoué, on reprend le recherche à zéro avec d' autres conditions.

Instanciation : association d' une valeur à un nom. FAIT : partie de la connaissance d' un système IA liée à un problème particulier. On distingue les FAITS des REGLES qui elles constituent une connaissance plus générale. Il y a souvent confusion des niveaux. Ainsi, en Prolog, le FAIT : (s,+,X,0,X). représente la "règle" de calcul : X+0=0 .

Filtrage (technique) (**pattern-matching**) : technique qui s' occupe de comparer une forme à un modèle. Exemple : Voici un filtre F = (mieux vaut X que Y). L' expression E = (mieux vaut tard que jamais) est filtrée par le filtre F. C' est à dire que la question : "Est-ce que F filtre E" aura pour réponse oui. En plus de répondre à la question, un filtre a souvent un effet de "bord" très important, celui de capturer les éléments indéterminés. Dans le cas de l' exemple X sera associé à "tard" et Y à "jamais". Diverses notations des variables de filtrage permettent de filtrer un mot, une partie de phrase, certains types de mot, etc.

Heuristique : une heuristique est une méthode, fondée souvent sur l' expérience et le jugement, qui est employée pour obtenir une solution à un problème.

Infèrence Symbolique : processus qui constitue les modes de raisonnement. Par exemple : le syllogisme. Certaines inférences prennent en compte des degrés d' incertitude.

Intelligence artificielle : secteur de l' informatique ayant trait aux concepts et aux méthodes pour représenter des connaissances symboliques et les traiter par inférences symboliques.

Lips : Nombre d' inférences logiques par seconde. Mesure de la vitesse des système d' IA. LISTE : structure de donnée qui consiste "simplement" (même si l' informatique sous-jacente peut être ardue) en l' énumération de certains objets. Exemples : la phrase "2 et 2 font 4" pourra se mettre sous la forme d' une liste : en Lisp : '(2 et 2 font 4) ; en Logo : [2 et 2 font 4] ; en Prolog : [2, et, 2, font, 4]

Modus Ponens (règle du) : c' est la règle qui, en logique des propositions ou des prédicats, permet de déduire q à partir de p et p$\Longrightarrow$q.

Prédicat : c' est un énoncé qui affirme une propriété d' un autre terme (sujet). De façon plus technique, c' est une fonction qui peut prendre pour valeur : vrai ou faux. Exemple : Le prédicat dans "Jean aime Marie" est aime. Par ailleurs, "aime" peut être considéré comme le nom d' une fonction à deux arguments : aime(Jean, Marie)

Règle : expression ou procédé qui relie entre eux diverses assertions.

Syllogisme : opération par laquelle, du rapport de deux termes avec un même troisième, on en conclut à leur rapport mutuel (Robert). Exemple : Tous les hommes sont mortels (prémisse majeure), Socrate est un homme (prémisse mineure), donc Socrate est mortel (conclusion). Par extension : tout raisonnement rigoureux, sans sous-entendu.

Système Expert (appelé aussi ASSISTANT INTELLIGENT ou SYSTEME A BASE DE CONNAISSANCES) : système informatique prototypique des systèmes conçus dans le domaine de l' IA. Les systèmes experts sont caractérisés par leur fonction (assistance), leur structure (base de connaissances et moteur d' inférences) et le type d' heuristique adopté (liée au fonctionnement d' un expert humain). unification : l' unification est une technique de filtrage qui permet l' utilisation de variable aussi bien dans le modèle que dans l' objet à "filtrer". (voir FILTRAGE) Exemples : (X vaut tard que Y) et (mieux vaut Z que Y) sont unifiables. X sera associé à "mieux", Z à "tard" et Y restera indéterminé. Par contre (X vaut tard que Y) et (mieux vaut X que Y) ne sont pas unifiables. En effet, X devrait être à la fois associé à "mieux" et à "tard".

Index