TE : Application de Prolog en IA en particulier les systemes experts

HARMACH Samia MAHE Sylvain
MARIEY Hervé

16 juin 2003

1 INTRODUCTION

Prolog est un langage qui permet un autre style de programmation que les langages les plus courants (comme C, Java ,Lisp...). Nous allons donc voir quelles applications ce langage permet de réaliser ,en particulier dans un secteur en plein développement : l'Intelligence Artificielle
L'Intelligence Artificielle ayant vu sa définition évoluer au fil années suivant les limites qu'on lui trouvait nous en donnons une ici :
L'I.A. est le domaine de l'informatique qui s'attache à essayer de faire en sorte qu'un ordinateur puisse élaborer un raisonnement tel que le ferait un être humain. Plus formellement, on parle d'I.A lorsqu'un algorithme non déterministe est utilisé, c'est à dire un algorithme susceptible de fournir des résultats différents à partir des mêmes paramètres d'entrée.[1]
Pour bien comprendre la spécificité du langage nous allons dans un premier temps en faire une brève description.

reférence : http://fr.wikipedia.org/wiki/intelligence_artificielle

2 BREVE DESCRIPTION DU LANGAGE

Prolog est le premier langage de programmation logique. Un programme Prolog est un ensemble de clauses appelées clauses de Horn (clause ayant au plus un littéral positif).Tout ce qui est écrit est vrai et tout ce qui n'est pas écrit est considéré comme faux par Prolog.Toute exécution d'un programme Prolog est une preuve. "Un programme logique est un ensemble d'axiomes ou de règles définissant des relations entre objets.Un calcul d'un programme logique est une déduction de conséquences à partir du programme.Un programme définit un ensemble de conséquences qui est sa signification. L'art de la programmation logique consiste en la construction de programmes concis et élégants qui ont la signification souhaitée."[2]. "Le résultat de l' exécution d'un programme Prolog est conséquence logique des axiomes qu'il contient."[2]. Prolog utilise

le système de chainage arrière c'est à dire : pour prouver une conclusion,on prouve les premisses de la règle qui amènent à cette conclusion.

Prolog à été crée en 1972 au GIA(Groupe d'Intelligence Artificielle) par Alain Colmerauer,Phillipe Roussel et Robert Kowalski à l' Université de Luminy à Marseille.Il a été inventé pour les développements en IA en particulier pour le traitement des langages naturels et pour les systèmes experts. Prolog est un langage de très haut niveau qui privilégie l' écriture et la lisibilité du programme au détriment de la rapidité du calcul(à l' époque de sa création).Mais ce n' est que dans les années 80 que Prolog est devenu populaire : grace à l' engouement des chercheurs pour l 'IA.[3]

"Bradley argues strongly that Prolog allows you to see more of the program on screen at any one time. The high level nature of the code means that the functionality of the process being encoded is expressed, without the usual morass of low level resource management routines. This he argues makes it easier to spot and prevent potential error. When errors do occur Bradley believes that these qualities make LPA Prolog easier to debug."[4] "It's much easier to develop and debug complex algorithms in this language."(Dr Gudes) [4]

7O % des applications en Intelligence Artificielle dans lŠindustrie se servent des Système Experts . Nous allons donc définir ce quŠest un système expert et voir comment on peut les implanter facilement en Prolog.

reférence: http://www.lim.univ-mrs.fr/ colmer/ArchivePublication/HistoireProlog

3 LES SYSTEMES EXPERTS

3.1 qu'est ce qu'un systeme expert?

Définition d' un système expert : Un système expert est un logiciel qui reproduit le comportement d'un expert humain accomplissant une tâche intellectuelle dans un domaine précis.

remarques :

-les SE sont généralement conçus pour résoudre des problèmes de classification ou de décision(diagnostic médical,régulation d' échanges boursiers,...).

-les SE sont des outils de l' IA c' est-à-dire qu' on ne les utilise que lorsqu' aucune méthode algorithmique exacte n' est disponible ou praticable.

-un SE n' est concevable que pour les domaines dans lesquels il existe des experts humains.Un expert est quelqu' un qui connaît un domaine et qui est plus ou moins capable de transmettre ce qu'il sait : ce n' est pas le cas d' un enfant par rapport à sa langue maternelle.

Un SE est composée de deux parties indépendantes : une base de connaissances elle-même composée de base de règles qui modélise la connaissance du domaine considéré et d' une base de fait qui contiant les informations concernant le cas que l'on est en train de traiter.

un moteur d' inférences capable de raisonner à partir des informations contenues dans la base de connaissance, de faire des déductions, etc.

Le rôle du cogniticien est de soutirer leurs connaissances aux experts du domaine et de traduire ces connaissances dans un formalisme se prêtant à un traitement automatique,c'est-à-dire en règles. Ces deux tâches sont aussi délicates l'une que l'autre. En effet, un expert est la plupart du temps inconscient de la majeure partie de son savoir ; et s'il arrive à en exprimer une partie, c'est souvent sous une forme qui ne se laisse pas faci-

lement formaliser. De plus l'extraction de connaissances peut avoir un effet perturbant sur l'expert : il est bien connu que si l'on demande à un maître aux échecs de réfléchir à sa manière de jouer, on observera une baisse de ses performances dans les jours ou les semaines qui suivent cet effort d'introspection.

L'indépendance entre la base de connaissances et le moteur d'inférences est un élément essentiel des systèmes experts. Elle permet une représentation des connaissances sous forme purement déclarative, c'est-à-dire sans lien avec la manière dont ces connaissances sont utilisées. L'avantage de ce type d'architecture est qu'il est possible de faire évoluer les connaissances du système sans avoir à agir sur le mécanisme de raisonnement. Il en est de même pour nous : un accroissement ou une modification de nos connaissances n'entraîne pas nécessairement une restructuration en profondeur de nos mécanismes de fonctionnement.

Dans la réalité, les choses se passent de manière un peu moins idéale et il est souvent nécessaire d'organiser la base de connaissances, de réfléchir sur les stratégies d'utilisation des règles, etc.

Le système expert est souvent complété par des interfaces plus ou moins riches permettant un dialogue avec les utilisateurs, l'idéal étant une interface en langage naturel.

3.2 La base de connaissances

3.2.1 La base de faits

2.1.1 Mémoire de travail

La base de faits est la mémoire de travail du système expert. Elle est variable au cours de l'exécution et vidée lorsque l'exécution se termine. Au début de la session, elle contient ce que l'on sait du cas examiné avant toute intervention du moteur d'inférences. Puis elle est complétée par les faits déduits par le moteur ou demandés à l'utilisateur.

Par exemple, dans le domaine médical, la base de faits pourra contenir une liste de symptômes en début de session et un diagnostic lorsque celle-ci se terminera.

2.1.2 Le type d'un fait

Les faits peuvent prendre des formes plus ou moins complexes. Nous n'envisagerons que des faits élémentaires dont les valeurs possibles sont

booléennes : vrai, faux symboliques : c'est-à-dire appartenant à un domaine fini de symboles réelles : pour représenter les faits continus. Par exemple, actif est un fait booléen, profession est un fait symbolique et rémunération est un fait réel.

Un système expert qui n'utilise que des faits booléens est dit d'ordre 0.

Un système expert qui utilise des faits symboliques ou réels, sans utiliser de variables, est d'ordre 0+.

Un système utilisant toute la puissance de la logique du premier ordre est d'ordre 1.

2.1.3 Les formules ou conditions

Dans un système expert d'ordre 0, on pourra par exemple écrire des formules de la forme : actif ou ň actif

Dans un système d'ordre 0+, on pourra trouver les formules :

actif et (profession ź medecin ou remuneration £ 20000)

Dans un système d'ordre 1, on pourra trouver :

+ Il exi X maladie(X) et X != grippe et symptome(X) = forteFievre

Ces formules sont appelées conditions lorsqu'elles servent à déclencher des règles. On remarque que les faits booléens peuvent être interprétés comme des formules puisqu'ils possèdent une valeur de vérité.

2.1.4 Métafaits et métavaleurs

Pour qu'un système expert puisse modéliser un raisonnement humain, il est indispensable qu'il puisse raisonner sur ses propres raisonnements, réfléchir aux faits qu'il manipule, aux formules qu'il peut construire, etc. Autrement dit, il n'est pas suffisant que le système ait des connaissances, il faut aussi qu'il ait des métaconnaissances.

Il faut par exemple qu'un système expert puisse savoir si une valeur a été attribuée à un fait. Dans la négative, cette valeur pourra être demandée à l'utilisateur. Mais si l'utilisateur ne peut pas répondre, il faudra que le système puisse le savoir afin de ne pas poser éternellement la même question. La seule manière d'attribuer une valeur à un tel fait sera alors de la déduire d'autres faits.

On dira que la valeur d'un fait est : connue si une valeur lui a été attribuée inconnue si aucune valeur ne lui a été attribuée et si aucune question à son sujet n'a été posée à l'utilisateur indéterminée si le système ne lui a attribué aucune valeur et et si l'utilisateur a répondu « je ne sais pas » à une question concernant sa valeur. Ainsi, valeur(profession) est un métafait symbolique et valeur(profession)= connue est une métacondition.

De manière analogue, tous les faits ne doivent pas faire l'objet d'une question à la personne concernée. Il n'est pas envisageable qu'un médecin demande à son patient « quelle maladie avez-vous ? » ni qu'un juge demande à la personne comparaissant devant lui « à quelle peine dois-je vous condamner ? »

Un système expert doit donc savoir si un fait est demandable ou non. Donc, demandable(diagnostic) est un métafait booléen (et une métacondition).

3.2.2 La base de règles

Elle rassemble la connaissance et le savoir-faire de l'expert. Elle n'évolue donc pas au cours d'une session de travail.

Une règle est de la forme Si <conjonction de conditions>alors <conclusion>

où les conclusions sont de la forme <Fait>= <valeur>.

Exemple : si population >200000 et ville-universitaire alors cinémaArtEtEssai=vrai

si revenu-imposable = connu et quotient-familial = connu alors calculerMontantImpot=vrai

Le dernier exemple montre comment un système expert peut être utilisé en association avec des programmes classiques. On peut supposer que le passage à la valeur vrai du fait booléen calculerMontantImpot déclenche une procédure calculant le montant de l'impôt en question et l'attribuant au fait réel MontantImpôt.

Une base de règles est un ensemble de règles et sa signification logique est la conjonction de la signification logique de chacune des règles. En particulier, on peut aisément coder dans le formalisme précédent des règles de la forme si A ou B alors C ou si A alors B et C Il n'en est par contre pas de même de si A alors B ou C

3.2.3 Les métarègles et la métaconnaissance

De même que dans un système expert, la connaissance est traduite en règles, la métaconnaissance s'exprime par des métarègles (c'est-à-dire des règles sur la manière d'utiliser les règles).

On trouve par exemple dans MYCIN les métarègles suivantes (voir [6]) :

si on recherche une thérapie alors considérer dans cet ordre les règles qui permettent de : acquérir les informations cliniques sur le patient trouver quels organismes, s'il en existe, sont causes de l'infection identifier les organismes les plus vraisemblables trouver tous les médicaments potentiellement utiles choisir les plus adaptés en plus petit nombre

si le patient est un hôte à risque et s'il existe des règles mentionnant des pseudomonias dans une prémisse et s'il existe des règles mentionnant des klebsiellas dans une prémisse alors il est probable qu'il faille utiliser les premières avant les secondes

si le site de la culture est non stérile et il existe des règles mentionnant dans leurs prémisses un organisme déjà rencontré auparavant chez le patient et qui peut être le même que celui dont on recherche l'identité alors il est sûr qu'aucune d'entre elles ne peut servir L'organisation d'une base de connaissances au moyen de méta-règles reste essentiellement déclarative, contrairement à toute organisation basée sur une structuration a priori de l'ensemble des règles (écrire les règles dans un ordre donné, ...).

3.2.4 La représentation des connaissances incertaines

Un des plus grand problèmes que rencontre le cogniticien lorsqu'il tente de formaliser le savoir d'un expert, c'est que celui-ci est capable de raisonner sur des connaissances incertaines et qu'on ne dispose que de très peu d'outils pour rendre compte de cette capacité.

Un médecin par exemple n'est jamais totalement sûr que tel symptôme est signe de tel maladie, que tel médicament sera supporté par le malade, que le malade guérira, etc.

On peut reconnaître globalement un objet sans être capable d'identifier à 100 pour cent chacun de ses détails.

L'observation du temps qu'il fait ne permet pas de savoir avec certitude s'il pleuvra ou non une heure plus tard.

On peut utiliser la théorie des probabilités pour définir le degré de vraisemblance d'un fait. De nombreux générateurs de systèmes experts offrent la possibilité aux utilisateurs de nuancer leur certitude concernant un fait en leur associant un degré de vraisemblance.

« Reconnaissez-vous votre agresseur sur cette photo ? »

« Oui, enfin c'est-à-dire, non. En fait, je le reconnais à 72

« Cette technique permet aussi de graduer une sensation.

« Avez-vous très mal à la tête ? Ou seulement, un peu mal »

« Sur une échelle qui associerait 0 au fait de ne pas avoir mal à la tête et 1 au fait de ne pas pouvoir le supporter, je dirai que j'ai mal à la tête avec un coefficient 0,45 »

« Les exemples qui précèdent révèlent une des principales difficultés de cette méthode : il n'est pas raisonnable d'attendre d'un être humain, expert ou non, qu'il puisse définir avec précision de tels degrés de vraisemblance.

« Un autre problème concerne la combinaison des coefficients de vraisemblance. Si les faits A et B sont connus respectivement avec les coefficients de vraisemblance p et q et si la règle si A et B alors C figure dans la base de règles, que peut-on déduire

pour le fait C ? Qu'il est vrai avec le degré de vraisemblance pq ? Mais ceci exige que les faits A et B soient indépendants. Comment s'en assurer ? Et si ils ne le sont pas ?

« Et comment rendre compte par une règle d'une connaissance du type

« le fait que A soit vrai rend B plus vraisemblable ?

« De nombreux travaux en Intelligence Artificielle sont consacrés à ces questions. On les regroupe généralement sous le vocable "recherche sur la logique naturelle" puisqu'il s'agit de comprendre en quoi nos raisonnements courants peuvent être décrits et formalisés par des règles logiques. On peut citer

« la logique floue qui explore la voie esquissée plus haut consistant à associer des coefficients de vraisemblances à des faits les logiques modales qui introduisent des opérateurs modaux comme "il est possible que" ou "il est nécessaire que" les logiques non monotones qui étudient la possibilité de réviser un raisonnement (si on me dit que TITI est un oiseau, j'en déduis qu'il vole mais si l'on ajoute que TITI est une autruche, je révise mon inférence).

« 3.3 Les moteurs d'inférences

Un moteur d'inférences est un mécanisme qui permet d'inférer des connaissances nouvelles à partir de la base de connaissances du système.

On distingue essentiellement trois modes principaux de fonctionnement des moteurs d'inférences :

le chaînage avant le chaînage arrière et le chaînage mixte On remarquera que les moteurs d'inférence décrits ci-dessous le sont indépendamment de tout domaine d'application. Cette séparation entre connaissance et raisonnement est essentielle pour les systèmes experts.

3.3.1 Le chaînage avant

Le mécanisme du chaînage avant est très simple : pour déduire un fait particulier, on déclenche les règles dont les prémisses sont connues jusqu'à ce que le fait à déduire soit également connu ou qu'aucune règle ne soit plus déclenchable.

Plus précisément, soit BF une base de faits, BR une base de règles (ne comportant que des faits booléens positifs) et Fait le fait que l'on cherche à établir, l'algorithme suivant calcule si Fait peut être déduit ou non de la base de connaissances.

ALGORITHME DU CHAINAGE AVANT ENTREE : BF, BR, F DEBUT TANT QUE F n'est pas dans BF ET QU'il existe dans BR une règle applicable FAIRE choisir une règle applicable R (étape de résolution de conflits, utilisation d'heuristiques, de métarègles) BR = BR - R (désactivation de R) BF = BF union concl(R) (déclenchement de la règle R, sa conclusion est rajoutée à la base de faits) FIN DU TANT QUE SI F appartient à BF ALORS F est établi SINON F n'est pas établi

On remarque que l'algorithme précédent n'indique pas comment choisir une règle applicable. C'est à ce niveau que la métaconnaissance du domaine intervient et permet de définir une stratégie de choix. Voir par exemple les métarègles implantées dans MYCIN au paragraphe 2.3.

Exemple :

Soit BF = B,C, Fait = H et BR composée des règles : Si B et D et E alors F Si G et D alors A Si C et F alors A Si B alors X Si D alors E Si X et A alors H Si C alors D Si X et C alors A Si X et B alors D

L'algorithme précédent appliqué à ces paramètres prouve que H se déduit de la base de connaissances.

Remarques : cet algorithme peut être très pénalisant pour certaines bases

l'algorithme de chaînage avant s'arrête toujours

si l'on utilise des règles dont les conclusions peuvent être des faits négatifs, pour tout fait F, il peut se produire 4 situations : F Î BF : le fait est établi. ň F Î BF : la négation du fait est établie. ni F, ni ň F ne sont dans BF : le système ne déduit rien à propos du fait. C'est une troisième valeur de vérité qui apparaît naturellement et dont l'interprétation peut être diverse selon les cas. F et ň F Î BF : la base est incohérente. On peut prévoir un fait Base_incohérente et une méta-règle de la forme : si il existe un fait qui appartient, ainsi que sa négation, à BF alors Base_incohérente.

le fait à établir peut ne pas être connu. Une étape de saturation de la base de connaissances consiste alors à déduire tous les faits déductibles

3.3.2 Chaînage arrière

Le mécanisme de chaînage arrière consiste à partir du fait que l'on souhaite établir, à rechercher toutes les règles qui concluent sur ce fait, à établir la liste des faits qu'il suffit de prouver pour qu'elles puissent se déclencher puis à appliquer récursivement le même mécanisme aux faits contenus dans ces listes.

L'algorithme de chaînage arrière est nettement plus compliqué que le précédent et nous nous contenterons d'étudier un exemple.

L'exécution de l'algorithme de chaînage arrière peut être décrit par un arbre dont les noeuds sont étiquetés soit par un fait, soit par un des deux mots et, ou. On parle d'arbre et-ou.

Si les faits déjà examinés ne peuvent pas être mémorisés (par exemple parce qu'ils sont trop nombreux), l'algorithme de chaînage arrière peut boucler.

On peut enrichir l'algorithme ce chaînage arrière en tenant compte du caractère demandable ou non d'un fait. Dans ce cas, lorsqu'un fait demandable n'a pas encore été établi, le système le demandera à l'utilisateur avant d'essayer de le déduire d'autres faits connus. Mais pour que ce mécanisme soit efficace (ce qui implique entre autres qu'il n'agace pas l'utilisateur en posant des questions stupides), il faut que le moteur d'inférences soit capable de déterminer quelles sont les questions pertinentes. Et ce problème est loin d'être simple.

Considérons par exemple la base de règles suivante : Si B et C alors A Si D et E alors A Si F et G alors A Si I et J alors G Si J alors ň E On suppose que les faits B, D, F et I sont les seuls faits demandables.

La mémoire de travail est initialisée avec l'information J est vrai.

La question posée au système est : A est-il vrai ? Quelles sont les questions pertinentes à poser à l'utilisateur ?

"B est-il vrai" n'est pas un question pertinente. En effet, aucune règle ne conclut sur le fait C qui n'est pas non plus demandable. Comme le fait B ne peut être utilisé que conjointement à C, la valeur de vérité de B n'apportera aucune information sur celle de A la question "D est-il vrai" n'est pas non plus pertinente. En effet, comme on sait que J est vrai, que cela implique que E est faux et que D n'est utilisé que conjointement à E, la valeur de vérité de D est inutile pour connaître celle de A la question "F est-il vrai" est pertinente. En effet, le fait G est encore déductible. Mais si la réponse à cette question est NON, la question "I est-il vrai" n'est plus pertinente car la valeur de G ne sert plus à rien.

3.3.3 Chaînage mixte

L'algorithme de chaînage mixte combine, comme son nom l'indique, les algorithmes de chaînage avant et de chaînage arrière. Son principe est le suivant :

Algorithme du chaînage mixte ENTREE : F (à déduire) DEBUT TANT QUE F n'est pas déduit mais peut encore l'être FAIRE Saturer la base de faits par chaînage AVANT (c'est-à-dire, déduire tout ce qui peut être déduit) Chercher quels sont les faits encore éventuellement déductibles Déterminer une question pertinente à poser à l'utilisateur et ajouter sa réponse à la base de faits FIN DU TANT QUE FIN

Bien entendu, les trois items figurant dans l'itération ci-dessus nécessiteraient d'être développés et précisés pour transformer ce qui précède en un véritable algorithme.

On trouvera dans les exercices une application du chaînage mixte.

3.4 conclusion

Les systèmes experts sont une des applications de l'intelligence artificielle qui ont quitté les laboratoires de recherche pour être utilisées dans le monde de l'entreprise. De nombreux systèmes experts ont été implantés avec succès pour résoudre des problèmes concrets. On les a même accusé d'avoir provoqué le crach boursier de 1986, ce qui est plutôt bon signe du point de vue de l'IA !

Mais les grandes difficultés que l'on rencontre lorsqu'on cherche à extraire des experts leur connaissance, puis lorsqu'on tente de formaliser ces connaissances dégagées à grand peine, sont peut être signes d'une faiblesse intrinsèque des systèmes experts. La possibilité de concevoir des systèmes experts repose en effet sur une hypothèse psychologique dont il n'a pas été question ci-dessus : la structuration de nos aptitudes en règles symboliques parfaitement identifiables. Majoritaire chez les psychologues et chercheurs en intelligence artificielle dans les années 70, cette hypothèse est de plus en plus battue en brèche et l'on envisage maintenant des modèles de savoir qui n'y ont plus recours (modèles connexionistes, systèmes adaptatifs, etc.). Il n'est par ailleurs pas exclu que les techniques à venir utilisent à la fois des systèmes experts et d'autres techniques comme les réseaux de neurones basées sur des hypothèses plus adaptatives que symboliques (systèmes hybrides).

4 LES APPLICATIONS PROLOG EN IA

Applications pour la recherche,compilateurs,analyse syntaxique et sémantique ,traitement du langage naturel, résolution d'enigmes...

4.1 traitement du langage naturel

Prolog permet le traitement du langage naturel c'est à dire comment interagir avec l' ordinateur en lui posant des questions dans la langue de celui qui l' utilise.(cf fichier "histoire Prolog" page 3-4). Par exemple,Eliza qui n'est ni un systeme expert ni vraiment de l'IA proprement dite, permet de simuler une conversation avec un psychologue :

« – exemple de conversation :
? -eliza.
Bonjour,Qu'est ce qui vous amene?
|:*je suis triste.*

depuis combien de temps etes vous triste?
|: *depuis le depart de mon frĄère*
pouvez vous me parler de votre frere ?
|: *j'aime discuter avec lui.*
quelqu'un d'autre dans votre famille aime t il cela ?
|:*non.*
continuez
|:*je me sens seule .*
ressentez vous souvent cela ?
|:itshape oui , depuis que mon frére est parti !
pouvez vous me parler de votre frere ?
|:*vous vous moquez de moi ?*
continuez
|:*bye*
Au revoir .J'espere vous avoir aider

« – principe du programme :
Reconnaitre une configuration de mots
Répondre en utilisant une configuration de réponses correspondantes
On définit donc des paires stimulus/réponse par exemple :
- je suis X
- depuis combien de temps vous etes X ?
On recherche des mots clés (père,mère , etc.) et on y associe une réponse.

« – algorithme :
Lire une phrase
Choisir une paire(stimulus,réponse)
Apparier la phrase et le stimulus
écrire la réponse associée
Recommencer

4.2 Analyse [7]

Une autre application de Prolog est l'analyse syntaxique et sémantique, qui n'est autre qu'une simple application des listes en Prolog.
Un analyseur consiste à reconnaitre la structure d'une information pour pouvoir lui donner un sens.Donc,étant donné une grammaire G (ensemble de régles) associée à un langage L ,on doit parvenir à construire un analyseur reconnaissant les mots de L(G).On fournira au programme une chaine sous forme de liste ,par exemple: 'a'.'a'.'c'.'b'.nil (il faut utiliser le caractère 'a' et pas simplement a qui est une variable), et on devra aboutir à un succĄés si la chaine est un mot de L(G), à un echec dans le cas contraire.
La construction d'un analyseur en Prolog ,peut se faire selon deux approches, une approche naturelle et une par les graphes.Ici on va se contenter de l'approche naturelle.
Soit la grammaire G =S $\rightarrow$ c;S $\rightarrow$ aSb
(L(G) = c , acb , aacbb , ...).

Les régles du programme doivent définir la structure des mots de L(G).
- S $\rightarrow$ c s'expime sans difficulté : une chaine est dérivable de S si cette chaine se réduit à 'c'.nil . Soit :

R1 derivable-deS(x) $\rightarrow$ se-reduit-a('c'.nil,x);
se-reduit-a signifie simplement que x est la chaine 'c'.nil.

-S $\rightarrow$ aSb est plus difficile à traduire. Il nous faut dire que la chaine d'entrée x se décompose en trois parties u , v ,w , avec les caractéristiques suivantes : u est le caractère "a' , v est une chaine qui dérive de S , et w est le caractère 'b' .La concaténation de u , v et w doit redonner x . Tout de suite ine difficulté apparait car on ne sait pas concaténer que des listes , il faut donc modifier u et w et prendre u = 'a'.nil et w='b'.nil . On pourra ainsi utiliser se-reduire-a introduit dans R1. On peut alors écrire:

R2 derivable-de-S(x) $\rightarrow$
se-reduit-a('a'.nil,u)
derivable-de-S(v)
se-reduit-a('b'.nil ,w)
concatener(y,w,x);

Il est à noter que ,puisqu'on ne peut concaténer que deux listes , il faut passer par une étape intermediare qui construit y .Il est alors possible d'utiliser la signification de se-reduire-a et remplacer u et w par leur valeur :

R1 derivable-de-S('c'.nil) $\rightarrow$;

R2 derivable-de-S(x) $\rightarrow$
derivable-de-S(v)
concatener('a'.nil ,v,y)
concatener(y,'b'.nil,x);

concaténer 'a'.nil à la liste v donne la liste "a'.v , on peut donc substituer 'a'.v à y . Attention, concaténer y à 'b'.nil ne donne pas y.'b'.nil qui est une liste dont le premier élément est toute la liste y . D'où la version finale
R1 derivable-de-S('c'.nil) $\rightarrow$;
R2 derivable-de-S(x) $\rightarrow$
derivable-de-S(v)
concatener('a'.v,'b'.nil,x);

Interpretation
- R1 : La chaine 'c'.nil est derivable de l'axiome.
- R2 : La chaine x est dérivable de l'axiome , si elle est la concaténation de la chaine 'a'.nil , d'une chaine v dérivable de l'axiome, et de la chaine 'b'.nil.

Ce programme est trés lent en exécution ,vu qu'il demande une concaténation dans une boucle de récursivité .Mais il n'y a pas d'autre amélioration évidente , il faut tenter une autre approche plus performante(approche par les graphes). Nous n'allons pas developper cette approche ,vu que ce n'est pas le but de notre exposé.

4.3 Résolution d'enigmes [2]

L'enigme logique consiste en quelques faits concernat un petit nombre d'objet ayant divers attributs. Le nombre minimum de faits est donné sur les objets et les attributs, afin de fournir un moyen unique d'assigner des attributs aux objets.
Voici un exemple qu'on utilisera pour décrire la technique de résolution d'enigmes lo-

gique.
Trois amis se retrouvent premier , second et troisième dans un concours de programmation .Chacun des trois a un prénom différent, aime un sport différent et est de nationnalité différente.
Michael aime le basket et a fait mieux que l'américain.Simon l'israélien , a fait mieux que le tennisman .Le joueur de cricket était premier.
Qui est l'australien? Quel est le sport pratiqué par Richard?
Des énigmes du type ci-dessus peuvent etre résolues avec élégance en instanciant les valeurs d'une structure de données convenable, et en tirant les valeurs et le troisième indice dans les conditions.
Chaque personne a trois attributs ,et peut etre représentée par la structure friend(Name,Country,Sport).Il y a trois amis dont l'ordre dans le concours est significatif.Ceci suggère une suite ordonné de trois éléments comme structure pour le problème , c'est à dire le liste.[friend(N1,C1,S1),friend(N2,C2,S2),friend(N

On se propose d'écrire d'autres programmes comme did-better , name , nationality, sport et first.
La combinaison de ces programmes marche comme une génération -et-test géante,chacun des buts did-better et member accèdent à des gens, et les autres buts accédent aux attributs des gens. Savoir s'ils sont générateurs ou testeurs dépend du fait que les arguments soient instanciés ou non.La réponse à l'énigme complète, pour les curieux, est que Michael est l'australien et Richard le tennisman.

4.4 Applications pour les entreprises aide à la décision pour les aéroports ,gestion des plannings [5]

4.4.1 applications

Modélisation de lŠenvironnement, Utilisé pour prédire le développement des forêts suivant les changements environnementaux, (Prolog a permis de vérifier les différents éléments composant le programme, il compare les différentes bases de données et demande à l'utilisateur ce qu'il faut faire sŠil existe des problèmes entre elles.)

4.4.2 ModélisatIdentification de champignon, de moisissure

Utilise pour prédire à l'avance l'espèce de champignon en train de croître, ce qui permet un gain de temps si ce n'est pas celle que l'on recherche. Prolog permet une comparaison avec la base de donnée déjà acquise sur les précédentes recherches.

4.4.3 traitement des salaires

Gérer par Ajax,un logiciel qui permet de faciliter la modification de la base de donnée, la gestion de d'emploi exceptionnel, Ajax étant programmé en partie en Prolog.

4.4.4 Modélisation des ventes

Machiaveli(système basé sur des règles) Le système modélise les relations entre les équipements, les gens et les demandes.Ce programme a été fait en Prolog car c'est plus simple a débuguer.

4.4.5 Entraînement des officiers des pompiers

(Iccarus) programme de simulation permettant l'aide à la décision pendant les grands incidents

"Cabinet de conseil"(InFlows) : Permet une meilleure utilisation du personnel.

"changement d'affectation"(Toranit)

4.4.6 Reconnaissance d'image

pas encore assez rapide, en développement, utilise Prolog+ une extension de prolog, pourrait être utilisé dans l'industrie pour le contrôle des robots, etc.

http://www.pdc.dk

5 CONCLUSION

Ce que l'on peut conclure sur les applications Prolog c' est que c' est un langage tres bien adapte pour l'IA et d' une maniere generale pour la recherche. Le deuxieme langage le plus utilise pour l' IA est LISP et tous ses dialectes.Ce sont les deux langages les plus utilises dans le monde pour toutes les applications en IA. Ce n'est pas un hasard car la manipulation de listes est tres bien adaptee pour l' IA et ces deux langages les utilisent abondamment.Cependant, Prolog reste LE langage des systems experts.

6 Bibliographie et lien Internet

[1] <http://fr.wikipedia.org/wiki/intelligence_artificielle>

[2] L'art de Prolog . Sterling, Shapiro .MASSON 1990 [3] www.lim.univ-mrs.fr/ colmer/ArchivePublication/HistoireProlog

[4] http://www.lpa.co.uk : Logic Programming Associates Ltd

[5] http://pdc.dk : Prolog Development Center

Elaboration dŠun système expert pour la conception de bases de données relationnelles à partir du modèle Entité-Assocation ,mémoire de Kolp Manuel (1993-1994).

[6]Un modèle concret dŠagent logique avec interface graphique, résolvant lŠindéterminisme par la réflexion, mémoire de Ian Rickrbush(2000-2001)

[7] Prolog . F.Giannesini, H.Kanoui ,R.Pasero , M. Van Caneghem . InterEdition 1985

[8] Programmer avec Scheme . J.Chazarin International Thomson Publishing 1996 Cette référence nous a servi à programmer un démonstrateur automatique de formules en ordre 0 en Prolog bien que le livre soit un livre sur Scheme.