CRIQUET Version 2. Un outil de base pour construire des systèmes experts

Philippe Vignard

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 380

CRIQUET
VERSION 2

UN OUTIL DE BASE POUR CONSTRUIRE DES SYSTÈMES EXPERTS

Philippe VIGNARD

Mars 1985

CRIQUET

Version 2

un outil de base pour construire des systèmes experts

Philippe VIGNARD +

+ INRIA, Sophia-Antipolis
06560 Valbonne FRANCE
Tel.: (93) 74-80-80
(poste 3305)

RESUME

L'objet de ce rapport est de décrire CRIQUET, un moteur d'inférence développé pour servir d'outil de base pour la construction de systèmes experts. Un module d'interface facilite la formulation des connaissances, spécifiées à l'aide de règles de production. Le système permet d'utiliser quatre types de raisonnements et des tests de compatibilité des faits et des règles. Les communications homme-machine sont facilitées.

ABSTRACT

This report describes CRIQUET, a production system for designing expert systems. A simple natural language understanding system facilitates the introduction and modification of production rules. Four types of search strategies including compatibility tests can be used. Of special interest are the man-machine communication facilities.

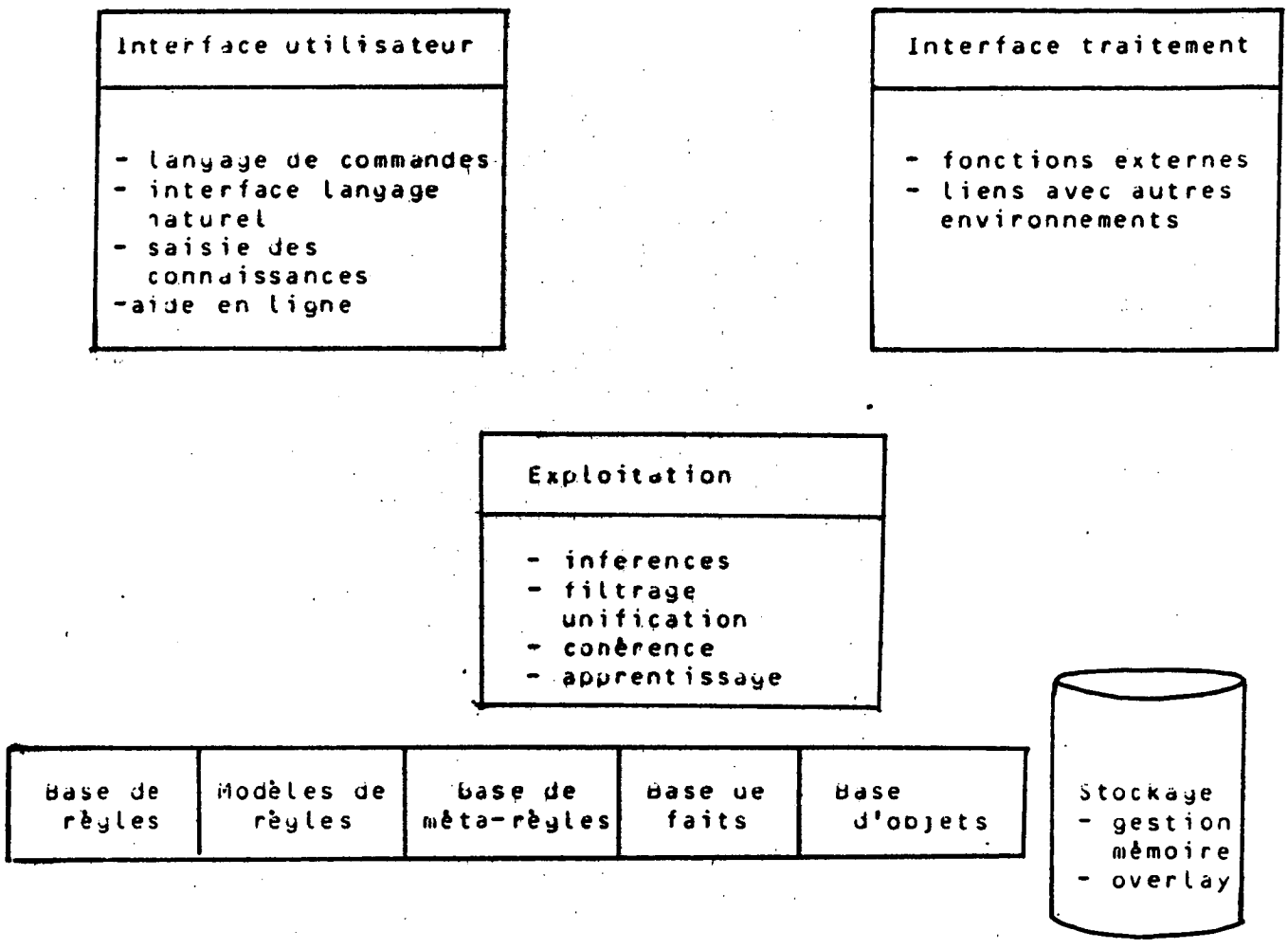
PLAN

La notion de systèmes experts recouvre de plus en plus de logiciels, dans de

nombreux domaines fort divers. La méthodologie des systèmes experts a déjà été appliquée avec succès, par exemple, pour le diagnostic médical (MYCIN), l'interprétation de spectrogrammes de masse (DENDRAL), la classification géologique (PROSPECTOR) ou la reconnaissance de la parole (HEARSAYII) (Waterman 78). Mais, un grand nombre d'utilisateurs potentiels sont intéressés en premier lieu, par une étude de faisabilité de l'approche système expert dans leur domaine. Pour cela, ils ont besoin d'un système facile d'utilisation, puissant, aisément modifiable et extensible, qui permette de construire le mécanisme de raisonnement adéquat selon le domaine.
CRIQUET est un système en constant développement, pouvant servir d'outil de base pour élaborer des systèmes experts dans divers domaines, où la modélisation des connaissances est possible à l'aide de règles de production.
Le moteur d'inférence CRIQUET a déjà été utilisé par des laboratoires publiques et privés. La version actuelle, écrite en Le_Lisp, constitue un ensemble de 400K au plus, soit environ 9000 lignes de Lisp. Des améliorations du logiciel sont prévues, ainsi que son transport sur micros ordinateurs, tels que IBM-PC et Macintosh.

I/ Structure générale du système

Le logiciel a été découpé en un grand nombre de ficniers, pour faciliter la lecture, la modifiabilité et l'extension du code. Ce dernier peut être utilisé comme un langage spécialisé de programmation pour élaborer le système adéquat. On peut aussi voir CRIQUET comme un système à part entière, composé de 5 grands blocs fonctionnels. Cette structure modulaire doit faciliter le développement et l'ajout même de fonctionnalités prévues mais non encore implémentées.



II/ Types de connaissances exprimables

Trois types de connaissances sont utiles et peuvent être exprimées avec le langage proposé :

- les connaissances sur le domaine, qui correspondent à l'expertise à propement parler spécifiée à l'aide de règles de production.

- la description du problème à traiter qui constitue un ensemble de connaissances volatiles sous forme de faits.

- des dépendances sémantiques, conditionnelles ou pas, entre les relations utilisées pour décrire le domaine et le problème.

La notion de contexte est utilisable et permet d'organiser la base de connaissances . Un contexte rassemble sous un nom donné les divers ensembles élémentaires de travail, tels que la base de règles, de méta-règles, la base de faits et d'autres ensembles. La base de connaissances peut être construite sous forme d'une hiérarchie de contextes, dans laquelle le système évolue en épuisant à chaque étape toutes les possibilités d'inférences dans le contexte utilisé. Les contextes peuvent être manipulés par les règles, c'est à dire dynamiquement en cours de raisonnement. Le système peut créer, détruire, modifier des contextes ou évoluer de l'un à l'autre.

III/ Le langage d'expression des connaissances

Le langage utilisé est de type calcul des prédicats du premier ordre. Des variables peuvent donc être utilisées dans l'expression des connaissances. Le langage est défini par un vocabulaire et une syntaxe :

1/ Le vocabulaire

Le vocabulaire comprend :
- des noms de prédicats d'arité 3 au plus.
- des noms de fonctions définies dans l'environnement de travail Lisp, pouvant référencer des procédures externes.
- des noms de variables commencant par ? : ?X, ?Y
- des noms de constantes.

2/ La syntaxe

Les termes du langage sont des quadruplets de la forme :

(objet prédicat attribut valeur)

L'utilisateur dispose d'un module d'interface pour l'aider à rentrer des connaissances selon un tel format. Ainsi la phrase "les isophotes sont de formes tourmentées" ou "la forme des isophotes est tourmentée" est traduite par le module en :

(isophote être forme tourmenté)

Le langage décrit ici permet l'utilisation de la négation. La phrase "les isophotes ne sont pas de formes tourmentées" est traduite en :

non(isophote être forme tourmenté)

Mais le quantificateur existentiel ($\exists$) et le connecteur ou ($\vee$) ne peuvent pas être utilisés.
A ou B --> C s'exprime par A --> C et B --> C mais A --> B ou C ne peut s'exprimer que par A et non B --> C et A et non C --> B ce qui complique les spécifications.
Des formats sont aussi à respecter dans la construction des règles et méta-règles.

3/ Termes pour l'appel de fonctions

Les appels de fonctions peuvent être utilisés dans les règles et méta-règles, en partie prémisses et actions. Un terme s'écrit alors :

terme := fonction : <appel>

<appel> = <nom-fonction> <liste arguments>

Les fonctions utilisées en partie conditions doivent rendre "t" ou "nil".

4/ Termes pour l'utilisation de prédicats

Des prédicats portant sur les variables peuvent être utilisés en partie prémisses des règles et méta-règles. Un terme s'écrit alors :

terme := prédicat : <texte>
<texte> = opérande1 opérateur opérande2

Les opérandes sont des nombres, variables ou atomes. Les opérateurs font partie de l'ensemble <, <=, =, <>, >=, > et "op-dans".

5/ Termes pour l'expression des actions dans les règles et méta-règles

Outre l'appel de fonctions, certaines actions sont propres aux règles ou aux méta-règles. Dans les règles, une action s'écrit :

terme := ajoutfait : texte /

:= suppresfait : texte /

:= suppresbut : texte .

texte = un quadruplet (objet prédicat atribut valeur)

On indique avec les mots clés "ajoutfait", "suppresfait" et "suppresbut" respectivement l'ajout ou la suppression d'un fait dans la base de faits, ou la suppression d'un but partiel dans la base de buts.
Dans les méta-règles, une action s'écrit :

terme := rht : mot /

:= rha : liste-mots /

:= rhp : liste-mots /

:= rhfa : texte /

:= rhfp : texte /

:= ajoutfait : texte .

Les "textes" ont la même définition que précédemment.
Les 5 premiers mots clés, définis dans les annexes, permettent des recherches sur divers critères parmi les règles objets. Le dernier, comme avec les règles, permet l'ajout d'un fait dans la base de faits.

IV/ Les constituants de la base de connaissances

Comme dans un système de production (Laurière 82), les 2 constituants classiques de la base de connaissances sont une base de règles et une base de faits. Les éléments composants ces bases sont construits à l'aide de termes du langage. Pour des facilités d'expressions et de manipulations, il est préférable de parler de base de connaissances du domaine (BCD) et de base de connaissances du problème (BCP). Chacune d'elles peut contenir des informations sous forme de faits et de règles (Cordier 84).
Dans le système CRIQUET, il y a une seule base de règles et base de faits courante. Mais il est possible de les sauvegarder dans des entités Lisp de noms donnés par l'utilisateur (commandes sauvbr,

sauvbf). Le nombre de ces entités est illimité. La restauration d'une base de règles ou de faits à partir d'une de ces entités dans la base courante est obtenue avec les commandes "chargbr" et "chargbf" en donnant le nom de l'entité à utiliser. Une base de faits ou de règles peut être construite par la réunion de plusieurs entités (ruf, rur). Ainsi une base de connaissances du domaine peut être constituée d'un ensemble de règles et de faits chargé lors de chaque référence au domaine. La base de connaissances du problème est constituée de règles et de faits que l'on rajoute aux bases courantes contenant la BCD.
Les commandes "brs" et "bfs" permettent de lister le contenu d'une entité Lisp utilisée pour sauvegarder des faits ou règles. La modification du contenu d'une telle entité ne peut être faite qu'en la chargeant dans la base courante concernée. Ces possibilités facilitent la mise au point d'une base de règles en permettant l'utilisation de plusieurs bases intermédiaires.

V/ Les faits et la base de faits

1/ Les faits

Ils peuvent être une connaissance du domaine ou une donnée particulière du problème et n'appartenir alors que temporairement à la base de faits. Un fait s'écrit avec un terme du langage, c'est à dire un quadruplet. Un coefficient de vraisemblance compris entre 0 et 1 est associé à chaque terme constituant un fait pour permettre un raisonnement approximatif. Un exemple de fait est :

((isophote être forme tourmenté) 0.8)

Le coefficient exprime l'importance du fait ou la confiance qu'a l'utilisateur en la réalisation de ce fait.

2/ La base de faits

Elle est constituée de faits et fournit les données particulières du problème. Différents utilitaires permettent de la gérer (voir annexes). Elle peut être construite par réunion de fichiers de faits . Des règles d'incohérence servent à assurer la cohérence sémantique des éléments la constituant .

```
CRIQUET  : bf
BASE  DE FAITS
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0
```

VI/ La base de règles

Des règles de production sont utilisées. Le moteur n'étant pas limité aux clauses de Horn, les règles peuvent avoir une ou plusieurs conclusions liées par un opérateur ET.
Les règles sont de la forme :

```
(type numero
      ( (condition-1)
        (condition-2)
            .
            .
        (condition-p))
      ( (action-1)
        (action-2)
            .
            .
        (action-q))
      (commentaires)
      (numeros de règles conseillées)
       coefficient)
```

où les (condition) sont des termes comme ils ont été définis précédemment. Les prédicats portant sur les variables permettent de spécifier des restrictions à respecter lors de la phase de filtrage.
Exemple :

```
si (isophote être forme tourmenté)
et ( T1 < ?X)
et ( T2 < ?Y)
et ( condition ( ?X < ?Y))
```

On a ici 2 variables (?X et ?Y) et une condition sur ces 2 variables.

Les (action) ont aussi été décrites dans un paragraphe précédent.
Les règles, comme les faits, sont pondérées d'un coefficient de plausibilité compris entre 0 et 1.
Le système est "ouvert" du fait de l'utilisation possible de règles entrainant le déclenchement d'un événement extérieur au système grâce à une action du type appel à une fonction. Avec les actions possibles pour les règles, divers effets de bords sont possibles.

L'utilisateur peut typer les règles. La distinction est spécifiée dans l'attribut "type" de chaque règle. Il peut ainsi partitionner la base de règles et organiser les connaissances comme il le désire. Cette organisation peut ensuite être utilisée à l'aide de méta-règles. Il existe par défaut 3 types de règles au choix : inférence, incohérence et équivalence. Dans une règle d'incohérence les conclusions se résument au mot "erreur". Ces règles permettent d'assurer la compatibilité sémantique, conditionnelle ou non, des faits.

Un exemple :

si (équation être type non-linéaire)
et (identification être méthode mco)
alors(erreur)

Si une règle est donnée du type équivalence, le système génère automatiquement la règle opposée : pour A --> B on ajoute B --> A. Le système manipule de même la contraposée des règles.

Les commentaires sont constitués de texte pouvant être consulté et utilisé par le système pour s'expliquer par la suite.

Les "numeros de règles conseillées" permettent au concepteur d'imposer des règles à utiliser après l'application de la règle en question, et ce lors d'un cheminement avant. Cet artifice améliore les performances du système, mais va à l'encontre de la notion de modularité de la base de règles et de "vrac absolu". Il faut donc faire attention lors de son utilisation.

Des utilitaires facilitent la gestion de la base de règles (voir annexes). La commande "ajr" permet l'ajout, en interactif, d'une nouvelle règle. Lors de l'utilisation de cette commande, le système maintient à jour l'ensemble des termes conclusion qui peuvent être utilisés comme but à diagnostiquer lors d'un cheminement arrière.

Exemples :

```
REGLE taxan NUMERO 7
SI ?animal etre mammifere
ET ?animal avoir sabot
ALORS ajoutfait (?animal etre ongule)
COEFF 0.8


REGLE taxan NUMERO 12
SI ?animal etre ongule
ET ?animal avoir rayure noir
ALORS ajoutfait (?animal etre zebre)
COEFF 0.6


REGLE coherence NUMERO 18
SI ?animal voler
ET ?animal etre autruche
ALORS erreur
COEFF 1.
```

VII/ Les autres ensembles de travail

Les 2 ensembles essentiels que nous venons de présenter sont la base de faits (bf) et la base de règles (br). Il y aussi une base de méta-règles (bmr), une base d'hypothèses (bh), une base de buts (bb), des contextes et un agenda courant.
Tous ces ensembles sont directement accessibles et modifiables par l'utilisateur.
La base d'hypothèses contient l'ensemble des termes reconnus par le système comme des diagnostics possibles. Elle est mise à jour automatiquement par le logiciel lors de l'ajout de règles.
La base de buts permet de spécifier des buts partiels utilisés, comme nous le verrons dans le chaînage mixte.
Un contexte permet de rassembler sous une même dénomination les 6 ensembles courants (bf, br, bmr, bh, bb et agenda). Les contextes peuvent aussi être manipulés par des règles dont l'activation permet une évolution dans une hiérarchie de contextes.
L'agenda permet la génération d'un plan de travail, puis son activation complète ou partielle et/ou sa mémorisation. N'importe quelle fonction Lisp peut être placée dans l'agenda. Celui-ci peut être manipulé par l'utilisateur ou par le système de façon dynamique.
Tous ces ensembles peuvent être sauvegardés plusieurs fois chacun. Des commandes permettent de restaurer une version mémorisée, et la gestion de ces différentes versions.

VIII/ Mécanisme d'exploitation

1/ Le filtrage et l'unification

Le filtrage consiste à mettre en correspondance 2 formes : le filtre et le fait. Seul le filtre peut contenir des variables. Ce processus permet de déterminer l'ensemble des conflits, constitué de l'ensemble des règles activables dans la situation courante.

Afin de faciliter l'écriture des règles, certains opérateurs de filtrage ont été définis :
- l'opérateur ? filtre n'importe quelle séquence de termes non vide
- les prédicats <, >, <=, >=, = appliqués à des nombres filtrent tout fait numériquement compatible au sens des inéquations.

Le motif (T < 3) filtre par exemple les faits (T = 0), (T < 1), (T <= -1).
L'algorithme utilisé dans la phase de pattern-matching, ou filtrage, est optimisé. Il ne procède pas par recherche exhaustive mais par essais successifs et retour arrière en cas d'échec.
Un processus d'unification a aussi été implémenté en particulier pour permettre des recherches parmi les règles à partir d'un terme pouvant

contenir des variables. Il offre les même possibilités de notation mais permet donc en plus l'utilisation de variables dans le filtre et le fait.

2/ Modes de raisonnement

Quatre types principaux de mécanismes d'inférence sont utilisés : le chaînage avant, le chaînage arrière comme dans la plupart des systèmes (Winston 81), le chaînage arrière partiel et le chaînage mixte.

a/ Le chaînage avant :

On cherche à déclencher les règles dont les prémisses sont vérifiées dans la base de faits. Une execution a toujours lieu suivant une séquence de cycles élémentaires, semblables les uns aux autres, dont la forme est :

```
-----------
I         I
I    détection : déterminer les règles et faits
I         I       pertinents par filtrage
I         I
I    choix      : choisir parmi les règles
I         I      activables celle à activer
I         I
I    déduction : exécuter les parties action
I         I       de la règle choisie
I         I
-----------
```

Le processus s'arrête quand on a atteint un but cherché ou si plus aucune action ne peut être déclenchée.
Le chaînage avant est déclenché par la commande "chav". Un but peut être spécifié au système avec des variables s'il y a lieu.

b/ Le chaînage arrière :

Le système tente alors de démontrer qu'un fait Z est vérifié en le considérant comme un but à atteindre. Une règle est applicable lorsque sa partie droite représente un sous but visé. L'appliquer consiste à demander de valider ses prémisses, au besoin en engendrant de nouveaux sous buts.
Lorsque le système n'a pas suffisament d'informations pour déclencher une règle et qu'aucun but ne peut être atteint, il interroge l'utilisateur et lui demande de valider un fait si possible.

le cheminement arrière est naturel si l'on a un ou plusieurs buts à vérifier. Il facilite la tâche de description du problème par les interrogations apportées sur des faits succeptibles d'affiner la description du problème traité.
Enfin, si l'utilisateur ne fournit pas de but particulier, le système dispose d'un ensemble de buts possibles et tente de les diagnostiquer séquentiellement jusqu'à la confirmation de l'un d'eux.
Le chaînage arrière est activé avec la commande "charr".

Exemple :

```
BASE DE FAITS
animal manger viande COEFF : 0.6
animal avoir couleur vive COEFF : 0.6
animal avoir tache noir COEFF : 0.5
sujet etre taxinomie animaux COEFF : 1.0

CRIQUET : charr
VOULEZ-VOUS PASSER PAR L INTERFACE LANGAGE NATUREL ? oui-non
? n
SI VOUS VOULEZ DIAGNOSTIQUER UN FAIT PARTICULIER
DONNEZ SON TEXTE
SINON RENVOYEZ " SEUL
? "
LE FAIT SUIVANT EST IL VRAI? oui-non
animal avoir poil
? ?
J'ESSAYE LA REGLE :
REGLE taxan NUMERO 1
SI animal avoir poil
ALORS ajoutfait (animal etre mammifere)
COEFF 0.5

VRAI ? oui-non
? oui
DONNEZ UN COEFF
? 0.8
LE FAIT SUIVANT EST IL VRAI? oui-non
animal donner lait
? non
IL EST VRAI QUE : animal etre guepard
```

c/ Le chaînage arrière partiel :

Ce processus est identique à celui du chaînage arrière exposé ci-dessus mais opère sans aucune interrogation vers l'utilisateur. Si le système ne parvient pas à valider un but de façon autonome, ce but est considéré faux et le système tente un autre diagnostic. Ce cheminement est voisin de la démarche utilisée en Prolog. Il est activé avec la commande "charr-p".

d/ Le chaînage mixte :

Un cheminement avant l'utilisateur ne connait pas le but à atteindre. Mais il peut dans certains cas donner au système une liste de buts intermédiaires. Ce sont des faits qui devront, selon l'avis de l'utilisateur, apparaître en cours de raisonnement.
Ces indications permettent de focaliser le raisonnement vers la résolution de buts donnés et ainsi de pratiquer un cheminement mixte, proche de ce qui est fait dans TANGO (Cordier 84). A chaque cycle on obtient en phase de détection un premier ensemble de règles candidates.
Le système utilise alors les buts partiels, en tenant compte ou pas de l'ordre dans lequel ils ont été donnés. L'utilisateur dispose d'une commande pour préciser ce point.
Dans le premier cas , le système valide si possible parmi les règles candidates celles qui contiennent en conclusion le but courant. Ce dernier est le premier dans la liste des buts partiels. Il n'est enlevé de la liste que lorsque une règle contenant l'action "suppresbut" est activée.
Dans le deuxième cas, le système travaille de la même façon mais en considérant tous les buts partiels restants. Un but partiel est éliminé s'il apparaît dans la base de faits.
Si aucune règle ne peut être ainsi validée le cycle se déroule en chaînage avant classique.
Cette technique permet de restreindre à certains moments l'espace de recherche et de réaliser des plans partiels de résolution à chaque cycle du moteur (Cordier 84).
Le chaînage mixte est utilisé à chaque appel du chaînage avant avec une base de buts non vide. Des utilitaires permettent de gérer la base de buts (voir annexes). Les commandes "avecbutord" et "sansbutord" permettent d'indiquer au système s'il faut respecter ou pas l'ordre dans lequel les buts partiels ont été donnés. La base de buts est gérée en mode LIFO. Le dernier but rentré lors de la création de la base sera le premier considéré par le système.

3/ Les types de recherches

Pour tous les types de cheminement proposés, l'utilisateur a le choix entre une recherche non exhaustive ou exhaustive.

Dans le premier cas, le backtrak (retour-arrière) est pratiqué automatiquement jusqu'à obtention du but donné s'il y en a un ou si une incohérence entre faits est décelée.
Dans le second cas, certaines contraintes sont à prendre en compte. Lors d'un cheminement avant, afin d'obtenir en dernier lieu une seule base de faits, on n'utilise à chaque cycle que les actions "ajoutfait" des règles. Sans cette restriction, à cause des possibilités de suppressions de faits, il faudrait pouvoir considérer une base de faits différente pour chaque parcours possible dans l'arborescence des règles. En chaînage arrière, on ne peut pas utiliser des variables libres, c'est à dire que les buts intermédiaires traités ne doivent pas contenir de variables. Un tel but partiel est considéré faux.

// Choix d'une règle

Il s'agit de choisir la règle à activer parmi l'ensemble des règles candidates et ainsi de résoudre les conflits existant si il y a lieu. La stratégie de choix est vitale non seulement pour les performances mais également pour la capacité du système à comprendre ce qu'il fait. Il faut déterminer les critères qui rendent une règle plus prioritaire que les autres.
Il existe trois grandes familles de structures de contrôle qui sont: choix par évaluation, recherche exhaustive et contrôle par méta-règles.
Dans le premier cas, une méthode est de prendre la première règle activable. C'est le critère le plus simple mais dépendant de l'ordre d'écriture des règles. On peut aussi choisir la règle la plus récemment utilisée ou encore la règle la plus contraignante, c'est à dire celle qui possède le plus grand nombre de termes condition. Dans ce système, 4 méthodes sont proposées :
- choix de la règle de plus forte vraisemblance
- choix de la règle utilisant les faits les plus récents
- choix de la règle qui utilise les faits les plus importants
- choix de la règle qui satisfait un but partiel

Dans le premier cas, on met l'accent sur les déductions. Dans le second, on favorise les objets et les transitions sont fonction des états et non uniquement des actions . De nouvelles fonctions de choix par évaluation peuvent être facilement introduites. La commande "chx" permet de passer d'une fonction à une autre.
Un contrôle par méta-règle peut aussi être utilisé comme dans MYCIN (Laurière 82, Winston 81). Il suffit pour cela de créer des méta-règles spécifiques qui permettent une sélection des règles objets selon leur type ou à l'aide de conditions portant sur les prémisses ou actions de ces règles. La phase de choix débute alors par une recherche d'une méta-règle activable. Celle-ci, si elle existe, fournit un critère de sélection parmi les règles objets candidates.

8/ Manipulation de la négation

Le filtrage sait reconnaître et manipuler la négation de termes. Le système n'opère dans un univers clos qu'avec l'approbation de l'utilisateur. Ce dernier peut ainsi infirmer un fait mais décider de ne pas mémoriser pour autant la négation de ce fait.

IX/ La méta-connaissance

La méta-connaissance est une donnée qui permet au système d'améliorer ses performances et son efficacité.

Pour améliorer de façon sensible les performances, en particulier dans le processus de filtrage , deux solutions particulières sont possibles: soit modifier le processus de filtrage (Cordier 84), soit réduire l'ensemble des règles à manipuler. Des méta-règles sont utilisées dans ce cas. Ces éléments introduits par R Davis (Davis 80) sont des règles qui agissent sur des règles objets. Les méta-règles font appel aux règles objet par leur contenu et non par leur nom. Ces outils ont déjà été utilisés dans beaucoup des systèmes tel MYCIN (Laurière 82)
Dans le système proposé des outils sont disponibles pour construire et utiliser de telles méta-règles. Celles-ci permettent de sélectionner ces règles par une recherche sur leur type ou sur un ou plusieurs mots ou termes dans les prémisses et conclusions (commandes "rht" "rhp" "rha" "rhfp" "rhfa").

Exemple d'une méta-règle :

```
CRIQUET  : bmr
REGLE meta NUMERO 1
SI sujet etre taxinomie animaux
ALORS rht (taxan)
COEFF 1.


REGLE meta NUMERO 2
SI ?animal avoir poil
ET ?animal manger viande
ALORS rha (?animal etre carnivore)
ET rhp (?animal etre carnivore)
ET rhp (?animal avoir poil)
COEFF 0.8
```

L'utilisateur peut indiquer avec les commandes "avecmeta" et "sansmeta" s'il veut travailler avec ou sans les méta-règles pour restreindre l'espace de recherches et ce, quelque soit le type de cheminement utilisé.

Les méta-règles peuvent aussi être intégrées aux structures de contrôle. Des heuristiques de guidage de la recherche agissent à travers ces méta-règles. Déja, l'utilisation de buts partiels, comme cela a été décrit précédemment, permet de guider les recherches du système et ainsi d'améliorer son efficacité. La fonction "chx" permet aussi d'obliger le système à utitiser un contrôle par méta-règles, en indiquant la fonction de choix par évaluation à utiliser dans le cas où aucune méta-règle ne peut être activée.

Ainsi, la structure de contrôle et les heuristiques sont mises en évidence au lieu d'être noyées dans le corps du programme (Davis 80).

4/ Le raisonnement approximatif

Les règles de production ne reflètent pas des implications logiques mais plutôt les convictions de l'expert.
Les règles comme les faits constituant l'univers sont valués avec un coefficient de vraisemblance compris entre 0 et 1.
Lors d'une déduction, la méthode de calcul d'une valuation pour un fait déduit est proche de celle utilisée dans MYCIN (Shortliffe 75).

Soit la règle R1 : SI A1 et A2 et ... An ALORS B le coefficient du fait déduit B est donné par :

$$cv(B) = cv(R1) * \text{Min } cv(Ai)$$

Nous n'avons pas voulu utiliser de concepts probabilistes du fait de la non vérification de certaines hypothèses mathématiques. Ainsi la non indépendance de termes condition de règles rend l'utilisation de tels concepts peu adaptée.
Il a déja été reconnu (Kayser) que l'utilisation privilégiée de max et min dans les fonctions de composition empêche tout effet de renforcement ou d'affaiblissement, alors que ces effets sont fondamentaux dans la représentation de l'évolution de notre jugement. Pour répondre en partie à cette critique un mécanisme de confirmation d'un fait a été mis en place. Si la règle activée confirme avec le coefficient cv2 un fait déja présent dans la base de faits avec coefficient cv1, le coefficient renforcé est alors égal à:
cv = cv1 + cv2 - cv1*cv2. Un tel mécanisme est déja utilisé dans des systèmes tels MYCIN et SNARK (Shortliffe 75, Lagrange 84) l'élaboration d'un mode performant de combinaison des coefficients n'est pas notre but et nous nous sommes contentés des mécanismes décrits ci-dessus. Dans le domaine de l'étude du raisonnement approximatif en général, de nombreux problèmes restent à résoudre (Kayser, Prade 82, Shortliffe 75).

XI/ Gestion de compatibilité des faits

Un raisonnement basé sur des coefficients de plausibilité peut empêcher de discerner, parmi les conclusions portant sur une même entité, les hypothèses concurrentes des composantes d'un même résultat. La gestion de compatibilité proposée permet de valider et d'affiner le raisonnement tenu par le système. Le système maintient, dans la limite de ses connaissances, la compatibilité logique et sémantique des faits.
La vérification de la compatibilité logique ne consiste qu'en la reconnaissance de la négation d'un fait lors d'ajouts de nouveaux faits. On ne peut pas ajouter la négation d'un fait déja présent. Il est possible ainsi de démontrer la négation d'un fait.
La compatibilité sémantique est assurée à l'aide de contraintes contextuelles spécifiées par l'utilisateur avec des règles d'incohérence proches des méthodes utilisées dans MIRLITHO (Ganascia 84)
La commande "coh" permet de spécifier une liste de faits toujours incompatibles avec un fait particulier. On donne pour un fait F1 une liste de faits (F2...Fn). Le système génère alors (n-1) règles d'incohérence telles que : si (F1 et F2) alors (erreur). Des incohérences conditionnelles peuvent aussi être données. Si les faits X et Y sont incompatibles en présence des faits A et B il suffit de créer une règle d'incohérence avec pour prémisse : (X et Y et A et B).

Cette gestion de cohérence est faite, si l'utilisateur le désire, lors d'ajouts de faits et lors de chaque phase de déduction dans le raisonnement.

Exemple : gestion à la création de la base de faits

```
CRIQUET  : avecgsem

CRIQUET  : prcoh
REGLES  DE COHERENCE :
REGLE coherence NUMERO 17
SI ?animal avoir poil
ET ?animal avoir plume
ALORS erreur
COEFF 1.

REGLE coherence NUMERO 18
SI ?animal voler
ET ?animal etre autruche
ALORS erreur
COEFF 1.


CRIQUET  : cbf
VOULEZ-VOUS  PASSER PAR L INTERFACE LANGAGE NATUREL ? oui-non
? o
```

```
POUR  SIGNALER LA FIN DES DONNEES TAPEZ " SEUL
DONNEZ  UN FAIT
? l'animal vole

PHRASE  CORRECTE VIS A VIS DU CONTEXTE
animal  voler
ETES-VOUS  D'ACCORD AVEC CETTE TRADUCTION ?
? o
DONNEZ  UN COEFF
? 0.8
DONNEZ  UN FAIT
? l'animal est une autruche

PHRASE  CORRECTE VIS A VIS DU CONTEXTE
animal  etre autruche
ETES-VOUS  D'ACCORD AVEC CETTE TRADUCTION ?
? o
attention  : ce fait est incoherent avec le reste de la BF
a  cause de la regle :
REGLE coherence NUMERO 18
SI animal voler
ET animal etre autruche
ALORS erreur
COEFF 1.

DONNEZ  UN FAIT
? "
VOILA  COMMENT J AI COMPRIS LA BASE DE FAITS
BASE  DE FAITS
animal  voler COEFF : 0.8
ETES  VOUS ACCORD ? oui-non
? o
```

Dans le moteur proposé, les règles d'incohérence sont valuées et manipulées comme les règles d'inférence. Le système peut ainsi tenir un raisonnement sans rejeter toutes les incohérences et ambiguités existantes. En effet, à l'aide de la valuation des règles le concepteur peut faire en sorte que certaines inférences soient possibles même en présence d'incohérences. Il suffit pour cela que les coefficients de vraisemblance de certaines règles d'inférence soient plus forts que ceux des règles d'inconérence candidates. Il est nécessaire alors de travailler avec la première fonction de choix citée précédemment. Ces possibilités sont intéressantes pour se rapprocher du raisonnement humain mais rendent très importante l'évaluation correcte des pondérations des règles.

On peut éprouver la validité de la base de règles en détectant des résultats contradictoires fournis sur un jeu de données valides (Ganascia 84)
Une règle d'incohérence activée signale l'incohérence et déclenche un retour-arrière .

XII/ Gestion de cohérence de la base de règles

Il est facile de garder une vue globale satisfaisante de l'ensemble des règles dans une petite base de connaissances. Mais quand on dépasse une certain seuil, avoisinant deux cent règles, des problèmes de cohérence apparaissent. La base de connaissances devient trop importante et très difficile à gérer sans outils spécialisés. Le problème de la cohérence se pose non seulement si plusieurs experts doivent enrichir la base de règles mais aussi lorqu'un seul s'occupe de la mettre en place. Les raisonnements spontanés du concepteur sont rarement totalement cohérents et sa mémoire insuffisante pour gérer une telle quantité d'informations.

Ce problème a déjà été pris en compte dans différents systèmes. TEIRESIAS offre un programme de "debugging " pour MYCIN afin de déceler des erreurs qui conduisent à des conclusions erronées. Mais on présuppose alors la règle douteuse et la base de faits correcte alors que le contraire peut se produire. ONCOCIN (Suwa 82) permet une analyse plus fine des problèmes de cohérence. Les tests pratiqués concernent la consistance logique et la complétude. La connaissance est représentée avec des règles de production et donc les vérifications réalisées se ramènent aux tests de conflit, de redondance, et de subsumption (spécialisation). Mais la vérification de la cohérence se fait de façon procédurale classique.

En s'inspirant de cela, la gestion de consistance logique dans CRIQUET est traitée de trois manières. Les tests codés permettent de déceler des cas de :

-conflit, quand deux règles sont sélectionnées dans la même situation mais aboutissent à des résultats conflictuels.

-redondance, quand deux règles sont sélectionnées dans la même situation et ont le même résultat.

-spécialisation, quand deux règles ont le même résultat mais que l'une a des prémisses contenant celles de l'autre.

Ces vérifications sur les règles objets sont faites à l'aide de méta-règles, telles qu'elles ont été définies prédemment. Par exemple, la méta-règle suivante suffit à elle seule pour gérer la redondance dans la base de règles :

```
REGLE meta NUMERO 1
SI regle ?x
ET regle ?y
ET fonction (equivalence (si ?x) (si ?y))
ET fonction (equivalence (alors ?x) (alors ?y))
ALORS ajoutfait (erreur de redondance)
ET fonction (print (numero ?x)(numero ?y))
```

La réalisation de la gestion de la base de règles avec des méta-règles laisse toutes libertés de manipulations à l'utilisateur.
Il peut décider de travailler avec ou sans contrôle. Un raisonnement peut être tenu sans rejeter toutes les incohérences qui peuvent apparaître. Le système fait remarquer à l'expert les problèmes de consistance logique mais seul l'expert décide en dernier lieu de la représentation des connaissances.
La gestion de la cohérence est modulaire. L'utilisateur peut définir des méta-règles en utilisant directement les "fonctions internes" du système. Parmi celles-ci, des fonctions prédéfinies concernent les notions de redondance et de conflit. La fonction "equivalence" utilisée dans l'exemple donné ci-dessus est un opérateur prédéfini qui permet d'établir la redondance de 2 listes de termes.
On peut aussi facilement élaborer de nouvelles méta-règles en modifiant ou en ajoutant de nouvelles fonctions internes Lisp dans un fichier spécial.
La gestion de la cohérence est donc implémentée de façon modifiable et modulaire.

XIII/ Manipulations de fonctions et interface avec d'autres environnements

Comme nous l'avons vu, des fonctions quelconques peuvent être utilisées dans les règles. On distingue les "fonctions internes" et les "fonctions externes". Les premières sont définies dans l'environnement Lisp et se contentent de celui-ci. Elles peuvent être des primitives de manipulations du système qui peut ainsi se modifier de façon autonome. Des fonctions internes prédéfinies permettent de simuler un langage arithmétique (Vignard 85).
Des "fonctions externes" sont des fonctions Lisp qui référencient des programmes externes, définis dans un tout autre environnement. Pour définir ces fonctions, l'utilisateur doit utiliser les "defextern" de Le_Lisp (Chailloux 83).
Cette dernière possibilité permet l'interface du système CRIQUET avec d'autres environnements tel que Fortran, Pascal. Des problèmes système sont à prendre en compte selon le système hôte et l'implémentation de Le_Lisp (Chailloux 83, Vignard 85).

XIV/ Compilation de la base de connaissances

Les accès à la base de règles selon un parcours totalement séquentiel sont une cause principale du manque de performance des systèmes de production classiques. Pour optimiser la recherche de l'ensemble de conflit et la mise à jour de la base de faits (ce qui peut occuper jusqu'à 80 % du temps de travail), on utilise une méthode qui consiste à établir des liens directs entre les règles et les termes par la

gestion d'un ensemble de fichiers. C'est ce que l'on entend par compilation de la base de règles.

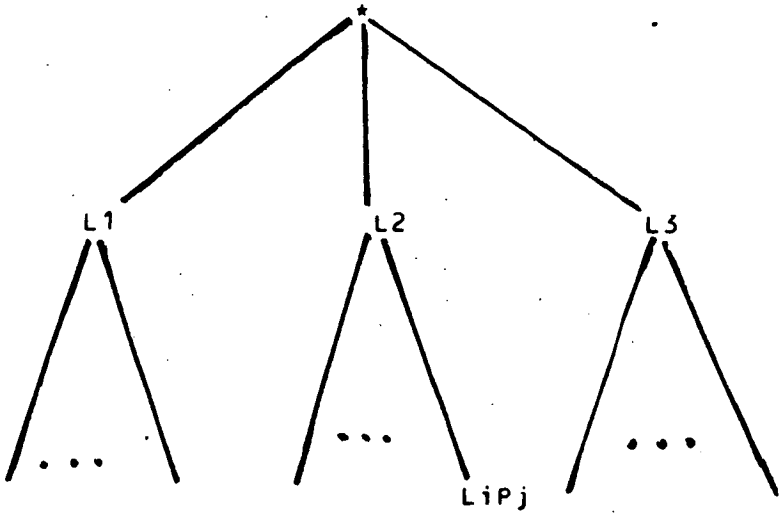
La technique de compilation implémentée dans CRIQUET s'inspire de diverses méthodes, dont le principe général est d'effectuer un prétraitement de la base, en tirant partie des recouvrements qui existent entre les règles (Cordier 84, Forgy 84).
Les idées de base sont tirées des constats suivants :

- la longueur des termes varie entre 2 et 4, et le prédicat n'est pas quantifiable. On peut donc partitionner l'ensemble des termes par 2 indexations successives, selon leur longueur et leur prédicat.

- les variables sont locales à chaque règle, aussi les termes qui ne diffèrent que par le nom de celles-ci sont normalisés en un seul terme. Par exemple, (toto aimer ?x ?z) et (toto aimer ?y ?t) sont représentés avec le terme normalisé (toto aimer $1 $2).

- un même terme apparaissant souvent dans plusieurs règles , des liens sont crées :
  règles ---> termes correspondants
  termes ---> règles les contenant

D'où une organisation générale des termes :



indexation selon la longueur

indexation selon le prédicat

Sur chacune des sous-listes LiPj ainsi constituées, on effectue la normalisation et la création des liens.

Organisation des éléments :

- liste des termes normalisés LTN = (T1, T2, ... Tn)
où Ti = (numéro, texte normalisé)

- liste des termes normalisés LiPj = (prédicat Pj, A1, A2 ... Ap)
où Aj = (numéro-terme-normalisé, R1, R2, ... Rn)
et Ri = (numéro-règle, compteur-C, liste-L) avec compteur-C = le compteur du nombre d'instanciations du terme dans cette règle, et liste-L = la liste des variables correspondantes aux variables normalisées dans l'ordre d'apparition.

- liste des règles LR = (R1', R2', ... Rm')
où Ri' = (numéro-règle, liste-L', compteur-C', (I1, I2, .. Iq))
avec liste-L' = la liste des variables intervenant dans les termes de la règle, et compteur-C' = le compteur du nombre de termes restant à instancier.
avec Ij = (liste des instanciations des variables apparaissant dans le terme). Le premier élément de chaque Ij est de la forme : (numéro-terme-normalisé, variables-correspondantes-aux-normalisées).

Une règle est supposée activable lorsque le compteur-C' est nul. Il subsiste alors le problème de résolution des jointures pour déterminer l'ensemble des substitutions complètes correctes pour chaque règle activable.

Cette compilation est réalisée de façon très modulaire. Le code généré constitue un module indépendant. L'utilisateur peut au choix compiler la base de connaissances, une fois qu'il l'estime correcte. Mais à chaque modification de la base, la compilation est annulée.
Grâce à ce procédé les performances doivent être nettement améliorées.

XV/ Communications homme-machine

Nous avons essayé de faciliter au mieux les communications homme-machine. Il est primordial que, quelque soit le domaine d'application, le système soit d'accès aisé aux utilisateurs, que ces derniers aient confiance et puissent vérifier les mécanismes de raisonnement utilisés. Ceci est fondamental à la fois dans la phase d'utilisation et de développement du système.

Un langage de commandes facilite l'accès au système. Toutes les commandes sont sans paramètres. Les éléments nécessaires au bon fonctionnement sont demandés par le système au moment voulu. Une aide en ligne est disponible et permet d'obtenir des informations sur un sujet donné ou une commande particulière. Les ajouts et modifications de règles et de faits sont aisés. Les modifications peuvent se faire avec un éditeur pleine page.

Un module simple d'interface rend plus aisée la donnée des faits et des règles.

Comme la plupart des systèmes experts, CRIQUET peut expliquer son raisonnement, quelque soit le type de cheminement, avec la commande "expl". Différents niveaux de précisions sont utilisables dans les

explications. Par défaut, le système ne donne une trace que du raisonnement qui l'a mené au succès. On peut aussi avoir une explication complète, avec trace de tous les essais infructueux et des retours arrière. Dans tous les cas, on peut décider d'utiliser ou pas les commentaires attachés aux règles.

Lors d'un cheminement arrière, l'utilisateur peut demander la justification d'une question posée par le système ("?"), et répondre par le doute ("??").

Un travail en mode "trace" est toujours possible et utile en particulier pour la mise au point des règles. Sous ce mode de travail, lors de chaque phase de choix, le système expose à l'utilisateur quelles sont les règles activables et quel est son choix. L'utilisateur peut l'approuver ou le rejeter. Dans ce dernier cas le système repète une opération de choix sur les règles activables restantes.S'il n'en reste plus le système arrête son raisonnement. L'utilisateur peut aussi demander l'explication du raisonnement tenu jusqu'alors ("expl") ou reinitialiser le dernier cycle ("rappel") en changeant s'il le désire la fonction de choix à utiliser ("chx"), ou en ajoutant des éléments dans la base de faits ("ajf") ou en modifiant les coefficients de vraisemblance de certains ("muf"). Ces fonctionnalités sont utiles pour le développement des bases de connaissances et la mise au point d'une pondération correcte des règles.

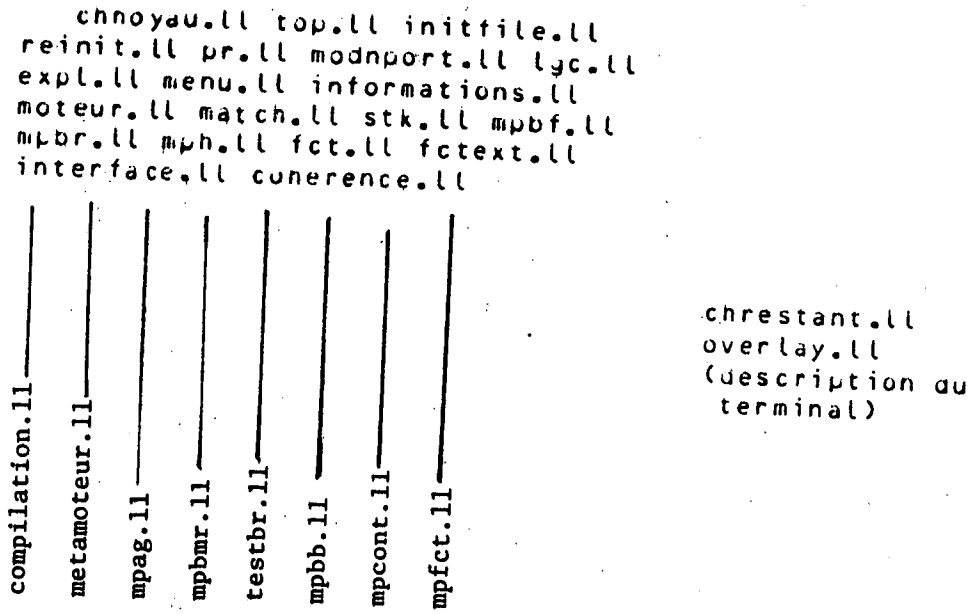
Dans tout travail d'élaboration d'un système la mémorisation d'un historique de travail est nécessaire. La commande "sauve" appelée en début de session permet, une fois sorti du système avec la commande "fin", de garder une trace de toutes les opérations faites durant la dernière session de travail. La commande "fin", au lieu de "end", permet de sortir du système en sauvegardant toutes les modifications faites dans les bases de règles et de faits. On peut aussi sauvegarder les ensembles de travail, tels que la base de faits, la base de règles ou d'autres ensembles, dans des fichiers du système hôte, que l'on pourra aussi recharger lors d'une autre session.

XVI/ Gestion mémoire

L'ensemble du logiciel a été découpé en environ 50 modules, organisés en hiérarchie. La racine est le noyau du système et doit être maintenue en mémoire. Les branches, constituées de modules indépendants représentent diverses fonctionnalités. L'ensemble est d'une taille d'environ 400K. La racine est de 100K. Cette structure permet l'overlay. A chaque appel à une commande située sur une branche, il faut charger le code porté par cette branche, après avoir effacé de la mémoire suffisamment de code pour libérer la place nécessaire. Ce mécanisme a été codé et est placé dans le module "overlay.ll". Des tables permettent de gérer les modules présents en mémoire vive. Une telle gestion de la mémoire peut être nécessaire

sur des micros ou des systèmes hôtes sans pagination.

Hiérarchie des modules



Les tailles données sont celles du code source, sans aucune optimisation. La place nécessaire peut facilement être réduite rien qu'en compactant les textes et en éliminant les commentaires.

XVII/ Portabilité

Le logiciel CRIQUET, rédigé en Le_Lisp, est portable partout où est implémenté Le_Lisp version 14 ou 15 (Chailloux 83). Des problèmes particuliers de transport peuvent provenir de la façon dont a été implémenté Le_lisp. Dans le logiciel CRIQUET, le code susceptible de causer des problèmes lors d'un transport a été regroupé dans un module spécial (modnport.ll). Des problèmes peuvent provenir de l'utilisation de fonctionnalités propres au système hôte (commande "comline" de Le_Lisp), ou des interfaces entre l'environnement Le_Lisp et un autre environnement (Vignard 85). Le logiciel a été déjà porté sur HB68, Vax 750 et 780 (VMS, Unix).

XVIII/ Modularité et extensibilité

CRIQUET peut aussi être vu comme un ensemble de primitives Lisp, faciles à combiner pour créer le système voulu. Au lieu d'un système, pour quelqu'un connaissant la programmation Lisp, CRIQUET devient un langage fonctionnel spécialisé de programmation. Le code a été commenté et découpé en petits modules pour faciliter une telle utilisation. CRIQUET, même vu comme un système, est facile à étendre et à modifier toujours grâce aux mêmes principes de modularité.

XIX/ L'apprentissage dans le système

Le logiciel permet aussi dans une moindre mesure l'apprentissage et l'auto-apprentissage du système. L'aspect fonctionnel du langage, la modularité et l'extensibilité du code permettent d'implémenter des tâches de ce genre. Le code, même s'il n'est pas sous une forme purement déclarative, est découpé en petits modules que le système lui-même peut manipuler et composer. Ainsi, à l'aide de méta-règles activant des fonctions, le système peut dynamiquement modifier la base de connaissances et les structures de contrôle. La pondération des règles objet par exemple, peut être dynamique, ce qui influe sur le contrôle et l'évolution du raisonnement. Avec quelques fonctions de plus, le système peut générer lui-même des règles. Mais, le but de ce travail n'est pas de réaliser un système avec apprentissage. Des outils sont offerts dans CRIQUET pour continuer l'étude dans cette direction.

XX/ Le module d'interface

1/ Introduction

Le module d'interface facilite la spécification des informations, en évitant à l'utilisateur d'avoir à formaliser ses connaissances selon le format interne strict utilisé par le système. L'interface tente de traduire une phrase simple en une représentation interne. Le vocabulaire employé dans les faits reste compatible avec celui des règles. Sans le module d'interface, des erreurs de spécifications sont possibles en particulier à cause des synonymes naturellement utilisés par l'homme mais méconnus du système. L'interface permet aussi d'enrichir le vocabulaire du système et ainsi sa connaissance.

2/ L'interface construite

Le traitement du langage naturel est un problème abordé en Intelligence Artificielle. Le but ici n'est pas aussi ambitieux. La grammaire reconnue est simple. Les phrases traitées doivent être suffisament courtes pour pouvoir être représentées à l'aide du format:

< objet prédicats attribut valeur >

Pour traiter chaque phrase, le système procède à une analyse lexicale, une analyse syntaxique et à des vérifications sémantiques.
Des dictionnaires de noms, de verbes et d'adjectifs sont utilisés. Des commandes permettent de compléter ou de modifier ces dictionnaires.
Pour chaque phrase, le système s'il ne décèle pas d'erreur propose sa traduction à l'utilisateur. Ce dernier peut s'il n'est pas d'accord reformuler son information.

3/ Principes de fonctionnement

La traduction se fait en deux passages. Le premier permet la suppression des mots vides tels que les articles, les adverbes et les prépositions. Le système dispose là aussi d'un dictionnaire de mots vides. Au deuxième passage, une analyse lexicale et syntaxique permettent de construire la représentation interne selon le format :

<nom><verbe><nom><adjectif>

3-1/ Analyse lexicale

L'analyse lexicale rend le type de chaque terme et le mot significatif associé. Il y a trois types de données reconnus : les noms, les verbes et les adjectifs. A chacun est associé un dictionnaire. Les dictionnaires sont des listes de la forme :

((mot-significatif type synonyme1 synonyme2 ...)(...) ...)

Le type des noms est : attribut ($a), objet ($o) ou neutre ($ao), celui des verbes est : transitif ou intransitif. Les adjectifs ne sont pas typés. Les mots significatifs sont ceux utilisés dans la représentation interne. Si un mot n'apparait dans aucun des dictionnaires, il est traité comme un mot inconnu.

3-2/ Analyse syntaxique

Deux classes de phrases sont reconnues en entrée, qui sont les formes actives et les formes passives. Le résultat de la traduction est toujours une forme active. Dans tous les cas, la grammaire reconnue est simple.

Forme active

- <nom><forme active transitive>((<nom><adj>)/(<adj><nom>)
exemple :"le chat mange la souris grise" est traduit en
(chat manger souris grise)

- <nom1><nom2><etre><adjectif>
Le complément du nom est traité et la traduction est :
(nom2 avoir nom1 adjectif)
exemple : "le chat du voisin est gris " est traduit en
(voisin avoir chat gris)

- <nom><forme active intransitive>
exemple : "mon voisin tombe " est traduit en
(voisin tomber)

Forme passive

- (<adjectif><nom>)/(<nom><adjectif>) <forme passive><nom>
exemple :"la petite souris est mangée par le chat" est traduit en
(chat manger souris petite)
Une forme passive est obligatoirement de la forme :
- <etre><verbe transitif>
La négation est reconnue par la présence simultanée des mots "ne" et "pas" ou par la présence d'un verbe préfixé par "n'" et du mot "pas".

3-3/ Vérifications sémantiques

Une analyse syntaxique qui s'achève correctement rend une expression de la forme : <nom1><verbe><nom2><adjectif>. Les éléments <nom1> et <nom2> possèdent un type, précisé dans le dictionnaire des noms, qui a été fixé par l'utilisateur selon le contexte de travail. Les trois types possibles sont : objet, attribut et neutre. Il y a erreur sémantique si la traduction rendue ne vérifie pas le format

<objet><prédicat><attribut><valeur>

c'est à dire si le premier nom n'est pas un objet ou neutre et le second nom n'est pas un attribut ou neutre.

4/ Enrichissement du vocabulaire et traitement des mots inconnus

Des commandes ("prd ajmd ajsd spmd mdm") permettent de modifier et de compléter les dictionnaires indépendamment de tout autre travail. Lors d'ajout de règles ou de faits, si les mots utilisés ne sont pas tous connus, CRIQUET dialogue avec l'utilisateur. Le système l'interroge pour apprendre le type du mot concerné, savoir si c'est un synonyme d'un mot significatif déjà connu ou pas. Les dictionnaires sont complétés comme il le faut selon les cas. Ce n'est que lors d'ajout de règles dans la base de connaissances que de nouveaux mots significatifs peuvent être introduits. De cette façon le vocabulaire des faits et des règles reste compatible.

Conclusion

Le moteur CRIQUET est en constant développement, en particulier en vue d'introduire une base d'objets structurés. L'introduction d'un mécanisme de filtrage flou doit aussi améliorer le processus de filtrage et permettre l'utilisation de la sémantique des objets. Le logiciel de base doit être amélioré afin de permettre la manipulation de la disjonction. Des problèmes systèmes sont à l'étude au sujet de la gestion mémoire et des possibilités de parallélisme. Le module d'interface peut aussi être amélioré en accroissant le nombre d'éléments qui composent les termes du langage et en élargissant la grammaire reconnue.

CRIQUET est élaboré au sein d'un projet plus important, EDORA (Gouze 84), développé au centre INRIA de Sophia Antipolis. Le sujet de ce projet concerne la modélisation des connaissances à l'aide d'objets dynamiques proches des frames, utilisant une dynamique à base de filtrage flou. Nous espérons introduire les résultats de ces recherches dans le logiciel CRIQUET.

Remerciements

Je tiens à remercier C.Ganyer, INRIA Sophia Antipolis, O.Corby, M.Montaldan, P.Reqivo en 1984 ainsi que A.Jouoerjean, S.Maure, d.Trousse en 1985, tous étudiants du DESS "Informatique et Sciences de l'Ingénieur", pour leur aide apportée dans le développement de ce moteur.

ANNEXES

Les commandes utilisables sont listées avec pour chacune une définition sommaire de leur fonctionnalité.

liste des commandes disponibles

Commandes du modules d'interface :
prd : pour imprimer le contenu d un dictionnaire du module d interface
ajmd : pour ajouter un mot dans un dictionnaire de l interface
spmd : pour supprimer un mot dans un dictionnaire
mdm : pour modifier le type d un nom ou verbe dans le dictionnaire associe
ajsd : pour ajouter un synonyme dans on dictionnaire
rchmd : recherche d'un mot dans un dictionnaire

Commandes de manipulations des faits :
ajf : ajouter des faits en conversationnel dans la base de faits
spf : supprimer un fait dans la base de faits
mdf : modifier le coeff de plausibilite d un fait
cbf : construire la base de faits
spbf : detruire la base de faits
bf : imprimer la base de faits
tribf : pour trier la base de faits sur le coeff de vraisemblance
rechbf : recherche associative dans la base de faits

Commandes de manipulations des hypotheses :
bh : imprimer l ensemble hypotheses
ajh : ajouter une hypothese
sph : supprimer une hypothese
spbh : supprimer la base d'hypotheses

Commandes de manipulations des buts :
ajb : pour ajouter un but dans buts
spb : pour supprimer un but dans buts
cbb : pour creer l ensemble buts
spbb : pour detruire tout buts
bb : pour imprimer buts

Commandes de manipulations des regles :
pr : impression d une regle
prl : impression d une liste de regles
prcoh : impression des regles de coherence connues
prc : impression de regles selon une recherche sur
sur le coefficient de vraisemblance
br : impression de la base de regles
mdr : pour modifier le texte d'une regle en edition de texte
mdrco : pour juste modifier le coeff de vraisemblance d'une regle
spr : suppression d une regle
ajr : ajout d une regle

rcp : pour generer la contraposee de regles
req : pour generer des regles equivalentes
spbr : supprimer toute la base de regles

coh : specifier une liste de faits incompatibles avec un fait
testbr : verifie la coherence la base de regles

compbr : pour compiler la base de connaissances

Commandes de manipulations des agendas :
sauvag : sauvegarder un agenda
chargag : charger un agenda
prlag : imprimer la liste des noms d'agendas utilises
ajag : ajouter un element dans l'agenda courant
spag : supprimer un element dans l'agenda courant
vidag : vider un agenda memorise ou le courant
extag : extraire un element de l'agenda courant
actag : activer un element de l'agenda courant, tout l'agenda
prag : impression de l'agenda courant

Commandes de manipulations des contextes :
crcont : creer un contexte
spcont : supprimer un contexte
prlcont : imprimer la liste des contextes crees
chcont : charger un contexte

Commandes de manipulations des fonctions :
ajfct : definir une nouvelle fonction interne ou en modifier une existante
spfct : supprimer une fonction interne.
prlfct : liste des fonctions internes creees

defext : definir une nouvelle fonction externe ou en modifier une existante
spfext : supprimer une fonction externe
prlfext : liste des fonctions externes creees

Commandes de recherches parmi les regles :
rhp : recherches de regles sur un ou plusieurs mots dans les premisses
rha : recherches de regles sur un ou plusieurs mots dans les actions
rhfa : recherche d une forme dans les parties actions
rhfp : recherche d une forme dans les premisses
rht : recherche des regles selon leur type
rhc : recherche des regles de coherence
rhi : recherche des regles d inference
rst : restriction de la base de regles utilisees

Commandes de manipulations des meta-regles :
pmr : impression d une meta-regle
bmr : impression de la base de meta-regles
ajmr : ajout d une meta-regle
spmr : suppression d une meta-regle
spbmr : suppression de toute la base de regles
mdmr : pour modifier une meta-regle

Commandes de manipulations des indicateurs :
avecmeta : pour travailler en utilisant les meta-regles
sansmeta : pour ne plus utiliser les meta-regles
avecbutord : pour tenir compte de l ordre des buts partiels
sansbutord : pour ne pas tenir compte de leur ordre
rchex : pour utiliser une recherche exhaustive
nrchex : pour ne plus utiliser de recherche exhaustive
avecglog : pour assurer la coherence logique lors des deductions
avecgsem : pour assurer la coherence semantique lors des deductions
sansglog : sans la gestion de coherence logique
sansgsem : sans la gestion de coherence semantique
avectrace : travailler avec traces
sanstrace : travailler sans traces - mode adopte par defaut
avecomment : pour utiliser les commentaires des regles
sanscomment : pour ne plus utiliser les commentaires des regles

Commandes pour activer un cheminement :
chav : activer le raisonnement en chainage avant
charr : activer le raisonnement en chainage arriere
charr-p : activer le raisonnement en chainage arriere partiel
chm : activer le raisonnement en chainage mixte

Commande pour choisir une structure de controle :
chx : pour choisir la strategie de choix

Commande pour avoir les explications du systeme :
expl : pour avoir les explications sur le raisonnement

Commandes de sauvegardes et chargements :
sauvbrc : sauvegarde automatique sans parametre de la BR courante
sauvbfc : sauvegarde automatique sans parametre de la BF courante
resbr : restaurer l ancienne base de regles
resbf : recupere la derniere base de faits

sauvbf : sauvegarder la base de faits courante
sauvbr : sauvegarder la base de regles courante
sauvbmr : sauvegarder la base de meta-regles courante
sauvbb : sauvegarder la base de buts courante
sauvbh : sauvegarder la base d'hypotheses courante

brs : pour lister le contenu d un fichier de regles
bfs : pour lister un fichier de faits

chargbf : charger une nouvelle base de faits
chargbr : charger une nouvelle base de regles
chargbmr : charger une nouvelle base de meta-regles
chargbb : charger une nouvelle base de buts
chargbh : charger une nouvelle base d'hypotheses

pnbf : lister les noms utilises pour sauvegarder des BF
pnbr : lister les noms utilises pour sauvegarder des BR
pnbmr : lister les noms utilises pour sauvegarder des BMR
pnbh : lister les noms utilises pour sauvegarder des bases d'hypotheses

pnbb : lister les noms utilises pour sauvegarder des bases de buts

Commandes diverses :
spnom : supprimer un nom dans l'une des listes citees
ajnom : ajouter un nom dans une des listes citees

sauver : avec 1 param, sauvegarde dans un fichier Multics
charger : avec 1 param, charger un fichier Multics
edit : avec 1 param, acces en edition pleine page sur un fichier Multics

ruf : pour faire la reunion d un fichier de faits avec la
base de faits courante
rur : pour faire la reunion d un fichier de regles avec la
base de regles courante , utilisable aussi avec des meta-regles

info : pour avoir des informations sur le systeme
help : pour avoir des informations sur une commande en particulier
sauve : pour garder une trace de toute la session
fin : pour sortir du systeme en sauvegardant l environnement
end : pour sortir du systeme sans rien sauvegarder
essai : pour avoir une petite base de regles pour des essais

Exemple : ajout d'une règle

```
CRIQUET  : ajr
VOULEZ-VOUS  PASSER PAR L INTERFACE LANGAGE NATUREL ? oui-non
? o
DONNEZ  LE TYPE DE REGLE ?
PAR  DEFAUT : i POUR INFERENCE
ip  POUR INFERENCE AVEC DES CONDITIONS SUR LES VARIABLES
c  POUR COHERENCE
? taxan
DONNER  UNE CONDITION, POUR FINIR " SEUL
? ?animal est un oiseau

PHRASE  CORRECTE VIS A VIS DU CONTEXTE
?animal  etre oiseau
ETES-VOUS  D'ACCORD AVEC CETTE TRADUCTION ?
? o
DONNER  UNE CONDITION, POUR FINIR " SEUL
? ?animal est maritime
LE  MOT SUIVANT EST INCONNU :

maritime

INDIQUER  SI C EST UN NOM <TAPER:nom>
OU  UN VERBE <TAPER:verbe>
OU  UN ADJECTIF <TAPER :adj>
OU  UNE FORME CONJUGUEE DU VERBE ETRE:<TAPER :etre>
OU  UN MOT VIDE < EX: ARTICLE ADVERBE PREPOSITION ETC>
DANS  CE CAS :<TAPER :ptmot>
SI  C EST AUTRE CHOSE:<TAPER :q>
? adj
EST  CE QUE CE MOT EST UN SYNONYME D UN DE CEUX QUI SUIVENT ?
bleu  chaud jaune gris carnivore pointu aigu ongule vive
noir long
? n

PHRASE  CORRECTE VIS A VIS DU CONTEXTE
?animal  etre maritime
ETES-VOUS  D'ACCORD AVEC CETTE TRADUCTION ?
? o
DONNER  UNE CONDITION, POUR FINIR " SEUL
? "
DONNER  UNE ACTION, POUR FINIR " SEUL
? ajoutfait : ?animal est une mouette
LE  MOT SUIVANT EST INCONNU :

mouette
```

```
INDIQUER  SI C EST UN NOM <TAPER:nom>
OU  UN VERBE <TAPER:verbe>
OU  UN ADJECTIF <TAPER :adj>
OU  UNE FORME CONJUGUEE DU VERBE ETRE:<TAPER :etre>
OU  UN MOT VIDE < EX: ARTICLE ADVERBE PREPOSITION ETC>
DANS  CE CAS :<TAPER :ptmot>
SI  C EST AUTRE CHOSE:<TAPER :q>
? nom
EST  CE QUE CE MOT EST UN SYNONYME D UN DE CEUX QUI SUIVENT ?
animal  lapin souris guepard pingouin autruche zebre girafe tigre
plume poil oeuf mammifere lait oiseau dent griffe regard viande
tache rayure jambe albatros sabot couleur
? n
EST  CE UN ATTRIBUT <TAPER : $a
UN  OBJET <TAPER : $o> OU
LES  DEUX <TAPER : $ao>
? $ao

PHRASE  CORRECTE VIS A VIS DU CONTEXTE
?animal  etre mouette
ETES-VOUS  D'ACCORD AVEC CETTE TRADUCTION ?
? o
DONNER  UNE ACTION, POUR FINIR ~ SEUL
? ~
DONNEZ DES COMMENTAIRES SI VOUS LE DESIREZ
? ~
DONNEZ DES REGLES CONSEILLES POUR LA SUITE
SI VOUS LE DESIREZ
? ~
DONNER  LE COEFFICIENT DE VRAISEMBLANCE
? 0.8

VOILA  COMMENT J AI COMPRIS CETTE REGLE

REGLE taxan NUMERO 19
SI ?animal etre maritime
ET ?animal etre oiseau
ALORS ajoutfait (?animal etre mouette)
COEFF 0.8

ETES  VOUS D ACCORD ?-oui--non
? o
VOULEZ-VOUS  CONTINUER ? oui-non
? n
```

Exemple : un cheminement avant en mode trace

```
CRIQUET  : chav
VOULEZ-VOUS  PASSER PAR L INTERFACE LANGAGE NATUREL ? oui-non
? n
SI  VOUS RECHERCHEZ UN BUT PARTICULIER
DONNEZ  SON TEXTE SINON RENVOYEZ "
? "
LES  REGLES ACTIVABLES SONT :
REGLE taxan NUMERO 1
SI animal avoir poil
ALORS ajoutfait (animal etre mammifere)
COEFF 0.5


REGLE taxan NUMERO 5
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9


AVEC  LA BASE DE FAITS :
BASE  DE FAITS
============================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

J'AI CHOISIT LA REGLE :
REGLE taxan NUMERO 5
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9

ETES  VOUS D ACCORD ? oui-non
? chx
INDIQUER  LA STRATEGIE A UTILISER 1. ou 2.
1.  : CHOISIR LA REGLE DE PLUS FORT COEFFICIENT
2.  : CHOISIR LA REGLE QUI UTILISE LES FAITS RECENTS
3.  : CHOISIR LA REGLE QUI UTILISE LES FAITS LES PLUS IMPORTANTS
4.  : CHOISIR LA REGLE QUI SATISFAIT UN BUT PARTIEL
5.  : CONTROLE PAR META-REGLES
VOUS  UTILISEZ EN CE MOMENT LA STRATEGIE NUMERO : 1.
? 2
LES  REGLES ACTIVABLES SONT :
REGLE taxan NUMERO 1
SI animal avoir poil
ALORS ajoutfait (animal etre mammifere)
COEFF 0.5
```

```
REGLE taxan NUMERO 5
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9

AVEC  LA BASE DE FAITS :
BASE  DE FAITS
============================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

J 'AI CHOISIT LA REGLE :
REGLE taxan NUMERO 1
SI animal avoir poil
ALORS ajoutfait (animal etre mammifere)
COEFF 0.5

ETES  VOUS D ACCORD ? oui-non
? o
LES  REGLES ACTIVABLES SONT :
REGLE taxan NUMERO 5
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9
AVEC  LA BASE DE FAITS :
BASE  DE FAITS
animal  etre mammifere COEFF : 0.4
============================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

J 'AI CHOISIT LA REGLE :
REGLE taxan NUMERO 5
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9

LES  REGLES ACTIVABLES SONT :
REGLE taxan NUMERO 9
SI animal etre mammifere
ET animal etre carnivore
ET animal avoir couleur vive
ET animal avoir tache noir
ALORS ajoutfait (animal etre guepard)
COEFF 0.6
```

```
AVEC  LA BASE DE FAITS :
BASE  DE FAITS
animal  etre carnivore COEFF : 0.54
animal  etre mammifere COEFF : 0.4
==========================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

J 'AI CHOISIT LA REGLE :
REGLE taxan NUMERO 9
SI animal etre mammifere
ET animal etre carnivore
ET animal avoir couleur vive
ET animal avoir tache noir
ALORS ajoutfait (animal etre guepard)
COEFF 0.6

ON  OBTIENT LA CONCLUSION SUIVANTE :
animal  etre guepard COEFF : 0.24
```

Exemple : une explication du système après un cheminement avant

```
CRIQUET  : expl
LE  SYSTEME VA EXPLIQUER SON RAISONNEMENT A L AIDE DES
REGLES  UTILISEES
LORS  D UN CHEMINEMENT AVANT
==========================================
AVEC  LA REGLE NUMERO 1
REGLE taxan identificateur (1.)
SI animal avoir poil
ALORS ajoutfait (animal etre mammifere)
COEFF 0.5


ET  LA BASE DE FAITS :
BASE  DE FAITS
============================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

NOUS  POUVONS DEDUIRE
ajoutfait  (animal etre mammifere)

--------------------------------
AVEC  LA REGLE NUMERO 5
REGLE taxan identificateur (5.)
SI animal manger viande
ALORS ajoutfait (animal etre carnivore)
COEFF 0.9


ET  LA BASE DE FAITS :
BASE  DE FAITS
animal  etre mammifere COEFF : 0.4
============================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1.0

NOUS  POUVONS DEDUIRE
ajoutfait  (animal etre carnivore)
```

```
---------------------------------
AVEC  LA REGLE NUMERO 9
REGLE taxan identificateur (9.)
SI animal etre mammifere
ET animal etre carnivore
ET animal avoir couleur vive
ET animal avoir tache noir
ALORS ajoutfait (animal etre guepard)
COEFF 0.6


ET  LA BASE DE FAITS :
BASE  DE FAITS
animal  etre carnivore COEFF : 0.54
animal  etre mammifere COEFF : 0.4
==========================
animal  avoir poil COEFF : 0.8
animal  manger viande COEFF : 0.6
animal  avoir couleur vive COEFF : 0.6
animal  avoir tache noir COEFF : 0.5
sujet  etre taxinomie animaux COEFF : 1,0

NOUS  POUVONS DEDUIRE
ajoutfait  (animal etre guepard)

---------------------------------
```

BIBLIOGRAPHIE

Chailloux J
"Le_Lisp de l'INRIA"
Doc INRIA Novembre 1983

Cordier MO, Rousset MC
"TANGO : moteur d'inférence pour un système expert avec variables "
Proc 4ième congrès AFCET-INRIA
Reconnaissances des formes et intelligence artificielle
25-27 janvier 1984

Cordier MO
"Les systèmes experts "
La Recherche n=151 Janvier 1984

Davis R
" Meta rules : reasoning about control "
Artificial intelligence journal
Dec 1980 -15- pp 179-222

Forgy C.L
"The OPS 83 Report"
Department of Computer Science
Carnegie-Mellon University
Report CMU-BC-84-133, 51 pages, May 1984

Ganascia JG
"Etude des contradictions entre les données dans les systèmes de diagnostic"
Proc 4ieme journees congres AFCET-INRIA
Reconnaissances des formes et intelligence artificielle
25-27 janvier 1984

Gouze JL, Vignard P
"EDORA : un système intelligent d'aide à la modélisation en biologie "
International 84 AMSE Conference "Modelling and Simulation"
Athen (Greece) June 27-29 1984

Granger C, Thonnat M, Vignard P
"Etude d'un système expert pour la classification de galaxies : SYGAL "
Les systèmes experts et leurs applications :
Journées d'étude et exposition
Palais des Congrès - Avignon 2, 3, 4 Mai 1984

Kayser D
"Vers une modélisation du raisonnement approximatif "
Proc Colloque "Représentation des connaissances et raisonnement dans les sciences de l'homme"
Saint-Maximin ed M Borillo pub par INRIA (440-457)

Lagrange MS, Renaud M
"Deux expériences de simulation du raisonnement en Archéologie au moyen d'un système expert : le système SNARK "
Les systèmes experts et leurs applications :
Journées d'étude et exposition
Palais des Congrès - Avignon 2, 3, 4 Mai 1984

Laurière JL
"Représentation et utilisation des connaissances.
Premiere partie: Les systèmes experts"
TSI Techniques et Sciences Informatiques Vol 1, n=1, 1982

Prade H
"Modèles mathématiques de l'imprécis et de l'incertain en vue d'applications au raisonnement naturel "
Thèse d etat , Toulouse III, Juin 1982

Shortliffe E, Buchanan BG
"A model of inexact reasoning in medecine "
Mathematical Biosciences 23, 351-379 (1975)

Suwa M, Scott A.C, Shortliffe E.H
"An approach to verifying completeness and consistency in a rule-based expert system"
AI Magazine Fall 1982 (pp 16-21)

Vignard Ph.
" CRIQUET : Manuel d'utilisateur "
Document interne INRIA 85 (31 pages)

Waterman D, Hayes-Roth (Eds)
"Pattern Directed Inference Systems"
Academic Press N.Y (1978)

Winston PH
"LISP"
Addison Wesley Publishing Company 1981

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique