LABORATOIRE D'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M.
Faculté des sciences Economiques

UNIVERSITE d'AIX-MARSEILLE II

ISSN 0291 - 5413

INFORMATIQUE FONDAMENTALE & APPLICATIONS

Dépositaire
B.U. Sc. Eco.
Aix-Mars. II

BULLETIN 29

SOMMAIRE

JUIN 1991

Adresse postale : FACULTé DES SCIENCES ECONOMIQUES
LITAM
14 rue Puvis de CHAVANNES 13001 MARSEILLE
91 90 13 20 P 420 et 421

EDITORIAL

E. Bianco

Informatique, la réalité dans l'irréel.

La petite voile blanche perdue au bord de l'horizon, et qui s'enfonce peu à peu dans le néant, le grand planeur sombre qui glisse sans bruit tout au long de la montagne, le grand char cahotant dans la poussière et la caillasse des chemins interminables, un passé pourtant proche, et qu'on ressent si lointain.

Dans la lumière vive qui irise la mer frissonnante sous le crin du mistral, d'énormes masses sombres semblent attendre. C'est d'ici que partit le grand Saint Louis, l'épée flamboyante de la civilisation au poing pour convaincre les infidèles. Et du royaume des infidèles nous reviennent les lourds vaisseaux foncés aux flancs chargés d'opprobre. Ces sinistres réservoirs qui s'en vont vider leur fange dans tous les caniveaux de la planète. Et la lèpre s'étend, la mer se couvre d'étranges plaques moirées qui lui rongent les poumons, sur les plages et les accores une immonde bave englue l'écume du ressac, de sinistres prédateurs prolifèrent comme une épidémie et s'attaquent à la chair même de l'océan. La terre n'est pas mieux partagée. De sombres amas de briques noircies au feu et à la crasse, des ferrailles tordues et des terrils de gravats s'envahissent de végétation maladive, traversés d'une population hâve et qui s'étiole au souvenir du vacarme perdu, du feu éteint à jamais, et de la fumée qui rendait l'air tangible et oppressant.

La maigre cheminée qui vomissait la suie et la puanteur a cédé la place à l'immense bobine qui s'orne d'un nuage. Le silence et le béton clair masquent un feu volé à l'enfer et que des sorciers tiennent par leurs incantations enfermé dans une prison dérisoire et qui, sournois, n'attend patiemment qu'une faille ...

Il n'est plus de forêt tranquille, à l'abri des montagnes de scories corrosives qui sont la véritable moisson des grands traités des peuples. Il n'est plus de vallée profonde enfouies de silence sous une neige lumineuse, à l'abri de ces toiles d'acier que l'araignée du profit tend pour saisir le touriste affamé de glissades.

Les Dieux grecs et latins parfois joyeux vivants, sont morts pour que la cité devienne nation. Le Dieu terrible et féroce qui leur a succédé, le Dieu sans nom pour qui tout est sacrifice, le Dieu source de toutes les calamités et qui se fait honorer pour sa bonté, agonise. Toujours plus sournois le successeur, son ancien allié, s'apprête à endosser la pourpre.

Les Esprits des lieux et des actes s'affolent, les djinns, les démons, les génies, les grands architectes de l'univers ou d'ailleurs, les elfes, les farfadets, les gnomes, les fées, les goules, les petits Pères des peuples et les grands, les korrigans, les lutins, les sylphes, les méduses et les

gorgones, les garamaoudes et les ondines s'agitent, l'Univers devient irréel. A qui rendre des hommages ? A l'ancien qui meurt tout doucement mais dont les sursauts sont meurtriers, à celui qui monte, sournois, dont le pouvoir s'étend et souffre de moins en moins le partage...

L'art devient délicat qui consiste à faire sa cour; De tout temps ces dangereuses périodes ont fait la fortune des Oracles. Les Madame Soleil et les Monsieur Lune abondent qui tracent leur route aveugle aux grands esprits directeurs tourbillonnants d'une tornade qu'ils ressentent géniale, bien souvent impitoyablement sacrifiés par leur Seigneur et Maître, sur l'hôtel de la Stupidité.

Oh certes, le vieux Dieu sénile a un tout petit peu plus jeune émule qui lui ressemble beaucoup et qui tente bien certains ... Mais si l'un est grabataire, l'autre a des béquilles et ses pieds ont trempé déja dedans la fange du pétrole. Le Dieu nouveau, lui, possède une nouvelle église ô combien plus solide, avec ses temples à New-York, Londres, Paris, Tokyo ... Ses arcanes se nourrissent au plus profond du Chiffre, et c'est le grand Ordinateur qui sert la Grand'Messe du nombre. Oyez ! jeunes générations et inclinez vous devant Moloch. Moloch parle l'Informatique, et vous lui chanterez vos prières en langage machine.

L'amour, les sentiments perturbent les bilans, la loi du bon profit souffre de peste émotionnelle, la Paix et le Bonheur règneront sur la terre quand tout s'exprimera, enfin, en termes de calcul.

STATUTS

E. Bianco

Résumé.

Dans le domaine de la représentation des données de nature hétéroclite, intervient la nécessité de noter dans le code image une information d'ordre géométrique qui permet aux systèmes et aux machines universelles de s'y retrouver dans l'étalement des valeurs en mémoire. Il faut également se donner des moyens pour traverser les diverses couches d'interprétation définies par les domaines d'application des différentes machines universelles ou dérouleurs. Et montrer enfin que la solution choisie permet de résoudre le problème posé.

STATUTS.

La **notion de statut** s'impose à partir du moment où il devient nécessaire de représenter en mémoire des objets de nature hétéroclite. Les nombres eux-mêmes peuvent tomber dans ce cas si l'échantillonnage des valeurs se place dans un vaste domaine. De tels problèmes se posent de toute évidence à cause du caractère fini et borné de la mémoire centrale, elle-même découpée généralement en tranches de dimensions identiques.

Pour gérer confortablement les représentations il devient commode de leur adjoindre une information supplémentaire qui est une sorte d'indication géométrique. On pourrait également adjoindre une indication sémantique qui préciserait la nature de l'algorithme capable de s'appliquer dessus. Je vais me limiter à la notion de statut.

Je prends la **Procédure Formelle Symbolique** comme support de raisonnement. Dans un contexte de **Champ Fictif** si je me réfère à la déclaration des files, **tableaux** ou **files dynamiques**, on constate que le, ou les statuts sont fournis dans la description de la file. Par exemple:

40 Tableau ident(8), type(1), dim(2), numéro(1) ;

10 000,1 000 file dynamique (h, l, t), 9(2(4), (6), n(3)),
3((1), (1), (1)),
1(p(2), (8)): R1, R2, XM ;

Les **variables de répétition** telles que "n" ou "p" permettent d'obtenir des statuts dont la structure est partiellement dynamique. De par le principe même de ce langage, la compilation d'une phrase de procédure formelle symbolique, doit fournir un code en deux parties, l'une qui correspond au **contexte**, et l'autre au **texte**. Le dérouleur d'un tel code prépare la configuration lors de la lecture du code du contexte, puis déroule alors le code du texte qui lui permet de faire évoluer la configuration.

La séparation du code programme en deux parties est indispensable pour le déroulement de **l'instruction d'intégration**. En effet, celle-ci a pour but d'amener un programme au contact direct du système. C'est par exemple, le cas de l'adjonction d'un **sous-système** supplémentaire. Or, cette opération d'intégration a précisément pour but de mettre le code du contexte à la portée du système, afin que celui-ci puisse précisément préparer une **tâche**. Cette opération de préparation de tâche est évidemment dévolue au système puisqu'il s'agit d'attribuer de la place en mémoire pour préparer un calcul.

A titre de remarque, on s'aperçoit que le rôle du dérouleur de système et de sous-systèmes, n'est pas tout-à-fait le même que celui de dérouleur de programme d'utilisateur, car les codes contextes ne sont pas localisés de la même manière. Dans le premier cas on trouve le code dans une file du système, dans l'autre le code contexte est en **file support**. Toujours à titre de remarque l'implémentation de l'ensemble peut se faire selon deux choix possibles: soit le dérouleur central ne déroule que du système et du sous-système, alors il faut un dérouleur logiciel pour un programme quelconque, sinon le dérouleur central doit jouer différemment selon l'état du déroulement. Il est possible de distinguer ces deux états à cause précisément de la théorie de l'autojectivité.

Problème.

La procédure formelle symbolique est un langage destiné à ramener sur le même niveau de traitement des opérations qui appartiennent à des niveaux conceptuels différents. C'est ainsi qu'un système (le programme qui constitue le système, celui-ci serait-il intégré) doit pouvoir observer sur un même plan à la fois le sous-système, et l' objet (programme ou donnée) que traite le sous-système. Il est donc nécessaire de se donner les moyens d'atteindre à ce résultat. C'est par le moyen de la machine universelle, en fait par un jeu de notions cohérentes du langage qu'elle traduit qu'on y parvient. Cette machine universelle se trouve confrontée notamment à une double situation:

a) Elle déroule normalement un programme, et il lui faut atteindre à une valeur de statut pour définir un opérande. Je dirai qu'il s'agit là d'une **lecture directe**.

b) On se situe par exemple, au niveau du système, quand celui-ci observe une file, construite par ailleurs, et qu'il doit explorer afin d'en charger une partie, ou bien d'en extraire une partie, dans le cadre des échanges. Le système doit pouvoir retrouver les statuts attachés à la file, et se ramener au cas précédent. Je dirai alors qu'il s'agit d'une **lecture indirecte**.

On saisit la différence entre les deux cas: en a), la machine universelle à la lecture du code contexte, prépare une configuration dans laquelle vont se retrouver notamment les valeurs des statuts dynamiques ou non. Lorsque la MU déroule le code texte elle trouve toutes les indications pour atteindre aux valeurs des statuts, afin de pouvoir atteindre ensuite aux valeurs des opérandes. N'oublions pas que c'est la référence qui sert d'indication de localisation et le statut d'indication de dimension.

Dans le cas b), la MU déroule ce programme qu'on appelle système, et son jeu est à cette occasion identique à ce qu'on vient de voir. Mais

lorsque le calcul amène à explorer une file contenue dans la file support, afin d'en extraire des données ou d'y inscrire des valeurs, il faut observer que la file en question est inconnue du système. J'entends par là que nulle part dans le code contexte du système je ne dispose des statuts de la file à explorer, qui a été construite tout-à-fait indépendamment du système et en tous cas ultérieurement.

Si la machine universelle était en train de dérouler du système, il lui faut se mettre à considérer un code que lui indique le système exactement au même niveau que le code du système lui-même. Cela ne peut pas se faire sans quelques précautions.

Il va falloir que la machine universelle puisse s'y retrouver, entendons par là qu'elle ne se mette pas à mélanger le code système et le code du programme observer, d'une part, et d'autre part qu'elle puisse trouver dans le code observé toutes indications utiles, et des indications de même nature que celles qu'elle traite dans un déroulement normal.

Le plus simple, conceptuellement parlant, et pour chaque image codée de file, consiste à noter intégralement toutes les indications de structures, cela sera évidemment redondant pour un déroulement normal, en lecture directe, mais indispensable pour un déroulement en lecture indirecte.

Remarque1.

J'ai l'intention de montrer, ici, l'existence d'une propriété. Peu m'importe en conséquence les critères, par exemple d'économie de place ou de rapidité. Ainsi, il est évident que normaliser comme je viens de dire la représentation des files, va coûter un peu de place. En effet, en déroulement direct, un programme qui est normalement "intégré" aura en mémoire une représentation unique, même si ce programme est en cours de déroulement pour le compte de plusieurs tâches différentes. En conséquence la machine universelle qui le déroule, pour une même file a besoin de deux sortes d'informations, une information de structure générale, en représentation unique, et des informations de structure locale telles que valeurs de variables de répétition pour une ligne déterminée, ou bien la relation qui existe entre une ligne créée et un statut. Ces informations locales doivent se trouver notées au niveau de la tâche. Tout simplement parce qu'elles dépendent directement du calcul.

Deux cas se présentent à moi, pour montrer simplement l'existence de la propriété, je me simplifie la vie en adoptant une représentation unique des structures de files. Et alors la machine universelle travaillera de la même manière en déroulement direct et en déroulement indirect. Mais il est évident que je perds de la place dans la mesure où toutes les structures générales, même pour les programmes intégrés, vont se trouver codées dans chaque occurrence de tâche.

Si je veux pouvoir récupérer cette place, il me faut alors concevoir deux façons de dérouler un programme, et je complique la machine universelle tout en restant dans le domaine du possible. N'oublions pas que la machine universelle doit pouvoir distinguer à tout moment les deux niveaux de déroulement direct et indirect.

De toute manière, quel que soit le choix, c'est la machine universelle qui va dérouler l'un et l'autre cas. Et c'est une instruction spéciale qui, lors de son déroulement, lui fournira l'information:

A1 := référence $[_{X,X}$, A

$[_{X,X}$ donne le numéro code de la référence dont il faut simuler le jeu, A est une référence interne dont la valeur est calculée à partir d'une référence normale de file support.

Remarque 2.

Cette instruction de référence, fait partie de cette sémantique qui assure la liaison entre plusieurs plans de complexité dans l'organisation du système. Elle sert à l'organisation des configurations. Au même titre que le triplet: insertion fractionnée, VIIF, insertion différée, quant à l'articulation des algorithmes.

Représentation des files.

En toute généralité, pour représenter une file dans un code, je me limite au cas de la file dynamique, de loin le plus compliqué, il me faut un ensemble d'informations que je résume dans la liste suivante:

(type) , (débordement) , (accès) , (références) , (index) , (statuts) , (variables de répétition) , (liste des variables de répétition) , (accès aux lignes) , (valeurs) .

Le **type** indique s'il s'agit d'un tableau, d'une file dynamique ou d'une file support, les codes pour un tableau seraient fortement réduits par rapport aux autres cas.

Le **débordement** indique que la valeur en cours d'affectation ne tient pas dans la place réservée à la file (notion de système).

Les **accès** sont des entiers au sens de successeurs pour des éléments de files, et qui permettent d'atteindre aux différentes rubriques: références, index, statuts, var. de rep. , val. des stat. , valeurs.

Les références sont les images des variables déclarées dans l'en-tête de file, pour l'exemple ci-dessus: R1, R2, XM.

Les **références** sont des jeux de valeurs qui permettent de connaître à chaque instant la valeur de chacune des références déclarées pour la file, telle qu'elle a été calculée auparavant. Dans ces valeurs se trouvent un accès à la file, un numéro de ligne, un nom de statut, un état.

Le numéro de ligne permet d'atteindre à la valeur de ligne par l'intermédiaire de l'accès aux lignes. Le nom de statut permet de connaître la structure de la ligne par l'intermédiaire de la liste de statuts et des valeurs de statuts. L'accès à la file permet d'atteindre le code complet de file objet.

Les **index** sont définis par une valeur propre, un numéro de référence à laquelle il est attaché et un état. La valeur de l'index définit l'élément dans la ligne, le numéro de référence permet de savoir la ligne et le statut.

Les **états** pour référence et index servent à savoir si ces variables ont été affectées au moins une fois.

les **statuts** comportent les codes des statuts déclarés dans la file, c'est ainsi que pour le statut de nom 1 dans l'exemple ci-dessus, on devra trouver au moins les informations suivantes: 2(0,2.1,8.x) . Le premier 2 indique 2 éléments dans le statut, le 0 indique une variable de répétition (ici p) et 2 le statut de l'élément, le couple 1,8 décrit le deuxième élément, et x représente toute valeur de p.

Les **variables de répétition** représentent les valeurs instantanées de ces variables telles qu'elles ont été déclarées dans la liste des statuts de la file.

la **liste des variables de répétition** contient autant de valeurs possibles pour toutes les variables déclarées qu'il y a de lignes possibles dans la file. Le nombre de lignes est représenté par le deuxième entier de l'en-tête de file dynamique.

L'**accès aux lignes** est une suite de triplets en nombre égal au nombre de lignes possible, chaque triplet indique l'accès à la valeur de la ligne dans la liste des valeurs de lignes, le nom de statut affecté à la ligne, et un état qui indique une valeur affectée ou non affectée.

Les **valeurs** sont accumulées dans la configuration dont le volume est indiqué par le premier entier de la déclaration de la file. Ces valeurs sont juxtaposées au fur et à mesure de leur création par le déroulement du programme.

Codage des références.

Je vais donc préparer le travail de la suite en choisissant un code pour les références, qui donne le moyen de traverser les différentes couches d'interprétation. J'appelle **couche d'interprétation** l'ensemble sémantique constitué par la machine universelle et le code qu'elle traite, subdivisé en: code algorithme, code configuration. Ce code algorithme peut également être le descriptif d'un algorithme qui ait la signification d'une autre machine universelle, on a donc apparition d'une nouvelle couche d'interprétation. Cela implique que la configuration elle-même se subdivise en deux parties, l'une qui a la signification d'un code d'algorithme, et l'autre, de la configuration qui lui est soumise. Ce

raisonnement peut encore s'appliquer à cette nouvelle configuration, ce qui ferait apparaître une autre couche supplémentaire d'interprétation. Dans tout ordinateur banal, on dénombre aisément quatre couches d'interprétation.

Fondamentalement, une valeur de référence c'est un numéro de ligne. Mais une référence est attachée à une file déterminée, par définition. Or, le problème que je soulève ici est sémantiquement double, je veux pouvoir faire apparaître une file, sous forme de code bien entendu, mais comme contenu d'une ligne. La conséquence en est que, vu d'un niveau d'interprétation supérieur il va me falloir rendre calculable l'attachement d'une référence, qui devra pouvoir passer d'une file à une ligne qui lui appartient et dont le contenu est à interpréter comme un code de file. Un petit raisonnement montre de quelle nature est la sémantique qu'il faut lier aux références, et donc comment structurer leur code. Une référence de file dynamique ou de tableau ne peut avoir qu'un seul rattachement possible, c'est celui de la file dans laquelle elle est déclarée. Il en est de même pour une référence normale de file support, tandis que la référence intérieure, elle, aura deux rattachements possibles, d'abord la file support, puis la file inscrite dans une de ses lignes. Dans tout dérouleur il y a des références qui doivent pouvoir disposer de trois rattachements possibles: la file du contexte du dérouleur qui est de la dimension d'une file support, puis un rattachement à toute file qui peut y être contenue dans l'une de ses lignes, et enfin si la file contenue est elle-même une file support alors il faut encore un rattachement à une ultime file contenue dans l'une de ses lignes.

Il devient alors évident que si le dérouleur est placé au même niveau que le système, c'est-à-dire ils sont construits avec un contexte commun, le dérouleur utilise la même file support que le système, et nous n'avons plus que deux degrés de rattachement pour les références.

Travail sur les files.

Le calcul sur une ligne se fait en plusieurs étapes. D'abord il faut calculer un numéro de ligne à partir de la valeur initial et en appliquant ensuite les opérateurs ← et → . On définit ensuite la ligne par une instruction ligne R1,h : 9 ;
qui fait intervenir la référence seule ou avec un index, et dans tous les cas le nom d'un statut, ici le statut 9. Et avant de déclarer cette ligne, si le statut en question contient des variables de répétition, il faut les avoir calculées. C'est le cas, ici pour la variable n.

Le déroulement de cette instruction ligne est en quelque sorte une déclaration, et le travail consiste à charger en file "accès aux lignes" , dans le triplet dont le numéro est le même que la valeur de la référence,

ici R1, d'abord la valeur de l'accès dans la file des valeurs à la première case de libre, ensuite le code qu'on a fait correspondre au nom du statut, lequel code permet en outre d'accéder dans la file "statuts" à la valeur actuelle du statut. Le troisième paramètre est un état de la ligne. Et si ce statut contient des variables de répétition leurs valeurs sont portées en "liste des variables de répétition" au même numéro de ligne que dans la file "d'accès aux lignes". Ce qui signifie que les deux files "accès aux lignes" et "liste des variables de répétition" se correspondent ligne à ligne. Cela tient au fait que dans une file dynamique à chaque ligne peut être attribué n'importe quel statut pris dans la liste de ceux qui sont déclarés en tête de la file.

Une instruction de calcul qui utilise un opérande qui fait appel à une ligne ou un élément de ligne d'une file exige dans son déroulement un calcul préalable de l'accès à la place de la valeur dans cette file, que ce soit pour y lire cette valeur, ou au contraire pour la mettre en place.

Je vais en conséquence décrire le travail de la machine universelle qui interprète le code d'un opérande qui aurait par exemple la forme de:

$[R1,h$ ou encore: $[R1$

dans le premier cas il faut atteindre la suite de cases qui doit contenir une valeur d'élément de ligne, dans l'autre cas il faut atteindre à la suite de cases qui doit contenir la ligne complète. Pour lire ou pour écrire dans l'emplacement il faut et il suffit d'en connaître l'accès et la dimension en nombre d'éléments minimum. Donc:

1) La valeur de la référence donne le numéro de ligne k,
2) k donne dans la file "accès aux lignes", l'accès à la ligne qui contient le triplet qui décrit la ligne t = v, s, e
3) s donne l'accès au code du statut dans la file "statuts", ici deux cas:
   - 31) le statut ne contient pas de variable de répétition, le code du statut donne la dimension de l'emplacement. Le calcul est terminé pour cet opérande.
   - 32) le statut contient au moins une variable de répétition, alors k donne dans la file "variables de répétition", l'accès à la ligne qui contient les valeurs des ces variables. Dans ce cas, le code du statut plus les valeurs des variables de répétition, donnent la dimension de l'emplacement. Et le calcul est alors terminé pour cet opérande.

Remarque 3.

Si le code d'une file, est lui-même enregistré sous forme d'une file dynamique, les rubriques: "types", "accès", "indice de place libre", "références", "index", "statuts", "variables de répétition", "liste des variables de répétition", "accès aux lignes", peuvent être notées sous forme de lignes, auquel cas un statut correspond à chacune de ces

rubriques. Le contenu de la ligne "accès" est alors une suite d'entiers ayant le sens de successeurs et qui permettent d'atteindre chacune des lignes suivantes: "références", "index", etc.

Pour la machine universelle qui veut explorer cette file, chacune des valeurs de "accès", qu'on désigne par ei, pour la commodité de l'exposé, sert à calculer une valeur de l'une de ses référence.

D'un autre point de vue, la file "accès aux lignes" contient dans chaque triplet un entier e2 qui permet d'atteindre à la valeur de la ligne de la file codée. En conséquence, toujours pour la machine universelle, atteindre une valeur de ligne revient à atteindre une ligne dont le numéro est la somme de e1+e2. Le e1 représente ici l'accès à la rubrique "valeurs". Je décris là, globalement le travail que doit réaliser la machine universelle quel que soit le type de câblage utilisé. Il suffit de constater que quelle que soit la méthode employée, l'information est suffisante pour s'y retrouver. Je vais essayer de montrer comment les choses se passent en utilisant les moyens du PFS. Il est alors évident qu'un algorithme, par exemple en procédure Formelle (ou autre) pourra faire le même travail.

DEMONSTRATION.

Je vais d'abord observer un phénomène important. Quand je décide de coder de manière unique l'image d'une file avec le descriptif de structure, la file des accès aux valeurs et les paquets de valeurs organisés par ligne, cela signifie que je me donne le moyen de pénétrer de deux façons différentes dans la file. Soit directement en me fixant un jeu de statuts qui me permet de voyager dans ces trois parties organisées en suite de lignes, soit de manière implicite en me donnant des moyens d'accéder directement aux structures. Mais dans ce second cas il me faut alors des instruments de programmation spécifiques, qui indiquent des opérations qui ne puissent se confondre avec les opérations standard.

Par exemple, et pour illustrer le premier cas, si je me réfère au paragraphe de la représentation des files, je peux me donner un statut qui décrive:

(type) , (débordement) , (accès)

par un statut qui permettrait d'en constituer une ligne:

1 (1(1) , 1(1) , 7(2))

et ainsi de suite pour les autres rubriques, de telle manière que si A est une référence qui indique la première ligne d'une telle file qui contient tous les codes, et si R une référence destinée à atteindre la première ligne de la suite (références), il suffit de calculer:

$$R := [_{A,3} \rightarrow A ,$$

Si I est une référence pour atteindre aux (index):

$$I := [_{A,4} \rightarrow A ,$$

si S est une référence pour atteindre aux (statuts):

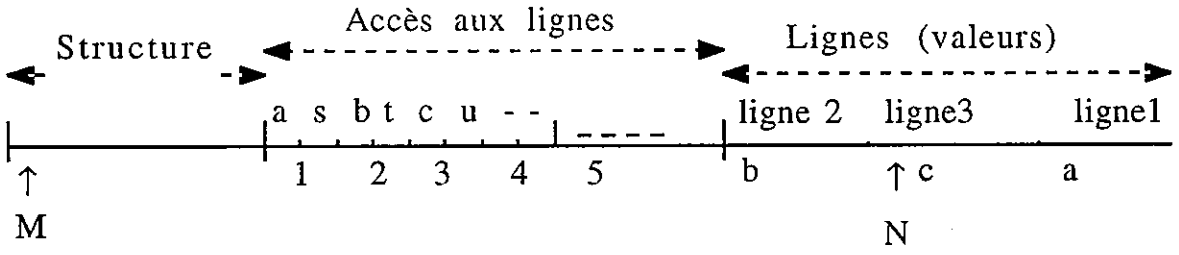
$S := [_{A,5} \rightarrow A$,

si D est une référence pour atteindre aux (accès aux lignes):

$D := [_{A,8} \rightarrow A$,

quels que soient les statuts choisis pour représenter ces diverses suites. Je définis ainsi un calcul direct pour les références. C'est le calcul de base en PFS.

Pour illustrer alors le second cas, je me donne un moyen d'obtenir un calcul indirect des références. Cela est utile pour indiquer que la référence doit se calculer en tenant compte du codage de la file observée. Le code de cette file est supporté par une **ligne de file support**, c'est une **référence interne**, donc déclarée comme telle, et initialisée à partir des valeurs de références ordinaires de file support. J'imagine deux références internes M et N, si je rassemble tout le code de la file en trois rubriques pour un exemple:



Si a, b et c sont les accès aux trois lignes de l'exemple, et s, t, u les statuts de ces lignes, alors N est calculé en deux temps à partir de la référence interne M:

I1 N := référence $[_{X,t}$, M

I2 $N := 2 \rightarrow N$

La sémantique que j'attribue à I1 dépend du codage choisi, je la définis ainsi:

Pour I1, la quantité que représente $[_{X,t}$, c'est-à-dire la référence codée, indique un numéro de référence à prendre dans le code de file indiqué par M. Est alors attribuée à N, comme valeur de référence, celle qui permet d'atteindre à la ligne indiquée par la référence codée. Une machine universelle qui déroule une telle instruction devra tenir compte des informations codées pour calculer la valeur de N à partir de ce que lui permet d'atteindre M. Le $[_{X,t}$ donne un numéro de référence, qui permet d'atteindre en file: (références), la valeur de cette référence au moment du calcul, il peut y avoir ou non un statut déja affecté. S'il y a déja statut affecté, c'est ce même statut qui est affecté à N.

Je vais me reposer les deux problèmes envisagés plus haut:

1) Comment une machine universelle calcule-t-elle ses propres références pour dérouler un programme dont elle consulte le code ?

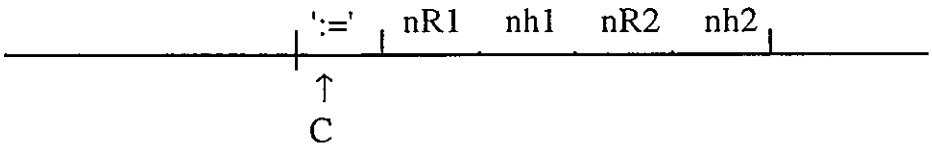
2) Comment travaille la machine universelle quand elle déroule un programme, qui est ce coup-ci un système qui a pour rôle de travailler sur une file objet. C'est-à-dire une file qui n'est pas sienne, déclarée dans son contexte, mais qui a été construite par un programme quelconque. Afin, par exemple d'en extraire une partie dans un but d'échanges.

PROBLEME 1.

Je me donne un exemple de programme quelconque destiné à être déroulé par la machine universelle. J'en isole une instruction particulière, que je suppose s'appliquer sur une file dynamique. La machine universelle dispose donc du code de l'instruction, au milieu du code complet du programme, et du code de la file, au milieu des codes de toutes les files du contexte de ce programme. Je suppose que la MU s'est calculé deux références pour atteindre à ces informations, A pour le code file, et C pour le code instruction. Les références A1 et A2 vont me servir pour les opérandes. Imaginons l'élément de code programme qui nous intéresse, image de l'instruction:

$[_{R1,h1} := [_{R2,h2}$

Code programme



si je suppose que les nR1, nR2, nh1, nh2 sont les codes des références et index R1, R2, h1, h2. Alors je peux calculer les références qui vont permettre à la MU d'atteindre directement les lignes de la file objet:

I1 A1 := référence $[_{C,2}$, A

I2 A2 := référence $[_{C,4}$, A

A partir de ce moment, les références A1 et A2 sont normalement affectées à la file objet et s'appliquent sur elle, comme toute référence définie dans le contexte de la MU s'appliquerait sur une file déclarée dans ce même contexte. En d'autres termes dès que ces deux instructions sont déroulées, les occurrences des identificateurs A1 et A2 permettent d'atteindre à toutes les propriétés de R1 et R2 notées dans l'image de la file sur laquelle elles s'appliquent.

PROBLEME 2.

Un système qui travaille dans sa file support, calcule une ou des références pour y atteindre une ligne qui contient normalement une image ou code de file construite par ailleurs. Il faut pouvoir pénétrer dans la file pour, par exemple, en extraire une ligne. On va analyser ce

qui se passe quand le système se déroule, qui cherche à atteindre une telle ligne. On a deux cas, 1) le système est un programme ordinaire, codé en mémoire. 2) le système est intégré, c'est-à-dire câblé au même niveau que la machine universelle.

1) Si je me place au niveau du système, pour atteindre à la file codée dans une ligne, je pars également d'une référence interne M, qui me permet de calculer M1 et M2, de la même manière que ci-dessus, avec une différence, toutefois, qui tient à la manière d'obtenir la désignation de la ligne à atteindre. Ci-dessus, la référence est une donnée qu'on récupère dans un code programme. Ici la référence est une donnée fournie par exemple par l'utilisateur qui veut qu'on lui liste telle ligne. Dans les deux cas il faut disposer d'une référence et qui ait une valeur attribuée. Pour faire le calcul j'imagine que je me sers de la première référence de la file objet (n'oublions pas que le système en ignore le nom), je la désignerai par son code "1":

J1 M1 := référence 1 , M

M1 est une référence interne du système mais prête pour atteindre à des lignes de la file objet.

Si je suppose que le système a calculé l'entier qui définit la valeur de la référence objet dans la variable:

$[_{X,x}$

alors:

J2 M1 := $[_{X,x}$ → initial

à partir de cet instant, M1, référence qui appartient au système, donc de code localisé en image de file support du système, est codée pour atteindre une ligne de la file objet dont le code est localisé dans une ligne de la file support.

En conclusion, si le système est construit au même niveau que la machine universelle (donc "intégré"), on est ramené à la même situation qu'au cas précédent puisqu'alors la file support est commune à ces deux organes, puisque déclarée dans un contexte commun. La question reste soulevée dans l'autre cas. La MU doit alors dérouler les codes des instructions J1 et J2. La MU possède sa propre file support, qui contient entre autres, l'image de la file support du système.

Pour J1, la MU se calcule une référence interne Z, à partir des valeurs de codes de M. Ensuite, la constante 1 désignant une référence, MU se donne le moyen d'atteindre à l'image de première référence dans la liste (**références**), cf. plus haut le codage de la file. Au moyen, par exemple d'une référence Q. L'interprétation de J2 consiste alors à attribuer une valeur à M1, qui, en la circonstance n'en a pas encore. Cette valeur rattache automatiquement la référence à la file d'accès aux lignes de la file objet, et lui définit une valeur de statut qui doit déja exister puisque la file a été construite par ailleurs. Bien sûr, ce raisonnement vaut moyennant la vérification d'existence de la ligne, ce

que permettent les codes prévus à cet usage dans la file d'accès aux lignes.

Si l'on admet comme code d'une référence, le quadruplet constitué par les valeurs:

AF , Val , statut, code aff

AF est une indication d'accès au code de la file, c'est cette valeur qui est modifiée lors du calcul représenté par I1, I2, J1.

Val est le numéro attribué par calcul à la référence:

Rx := initial donne la valeur: 1.

Rx := 3 → Rx donne la valeur: 4.

statut est un accès au statut attribué.

code aff indique si la référence a été calculée.

Les systèmes experts et la machine universelle

Par

Ennacer BOUDIBA

Mots clés :
Intelligence Artificielle , Systèmes experts , Génie logiciel , Génie cognitif

Résumé :

Les systèmes experts sont des programmes particuliers, séparant nettement les connaissances spécifiques du domaine étudié de la partie procédurale chargée d'utiliser ces connaissances. Leurs utilisations dans notre monde réel permet une aide à la réflexion, une aide à la décision, une aide au pilotage opérationnel et une aide à l'action immédiate. Le but de cet article est de donner aux lecteurs les fondements de base des systèmes experts et les outils nécessaires pour leur développement .

Apparition des systèmes experts

Après la seconde guerre mondiale , des groupes de chercheurs travaillaient sur le développement des machines électroniques capables d'effectuer des calculs numériques complexes . **Alan TURING*** , affirmait qu'une machine conçue à cet effet pourrait avoir d'innombrables applications différentes .
Alan TURING prétendait que les instructions à fournir à une telle machine devraient s'appuyer sur des opérateurs logiques (ou,et) , qui sont des opérateurs généraux et de ces opérateurs on pourrait utiliser des opérateurs numériques spéciaux .Les programmes qui utilisent des opérateurs logiques seraient capables de manipuler des connaissances symboliques avec lesquels on voudrait travailler. Ainsi les chercheurs qui sont d'accord sur l'idée de TURING travaillaient en même temps sur les opérateurs non numériques ,pour permettre de concevoir une machine capable elle aussi d'interpréter les données utilisant les opérateurs non numériques connus sous le nom d'opérateurs logiques. Un autre travail se faisait en parallèles par les psychologues qui s'intéressaient aux problèmes humain et qui cherchaient à développer des programmes informatiques pour **simuler le comportement humain** c'est à dire comment il raisonne , comment il marche et se déplace et comment il parle et lit. A partir des deux (02) domaines de recherche qui sont le traitement symbolique et la résolution des problèmes humains, est engendré un champs interdisciplinaire de l'informatique que l'on appelle intelligence artificielle .
Pendant la période des années 1980-1985 , le domaine de l'intelligence artificielle est subdivisé en trois (03) sous domaines qui sont :

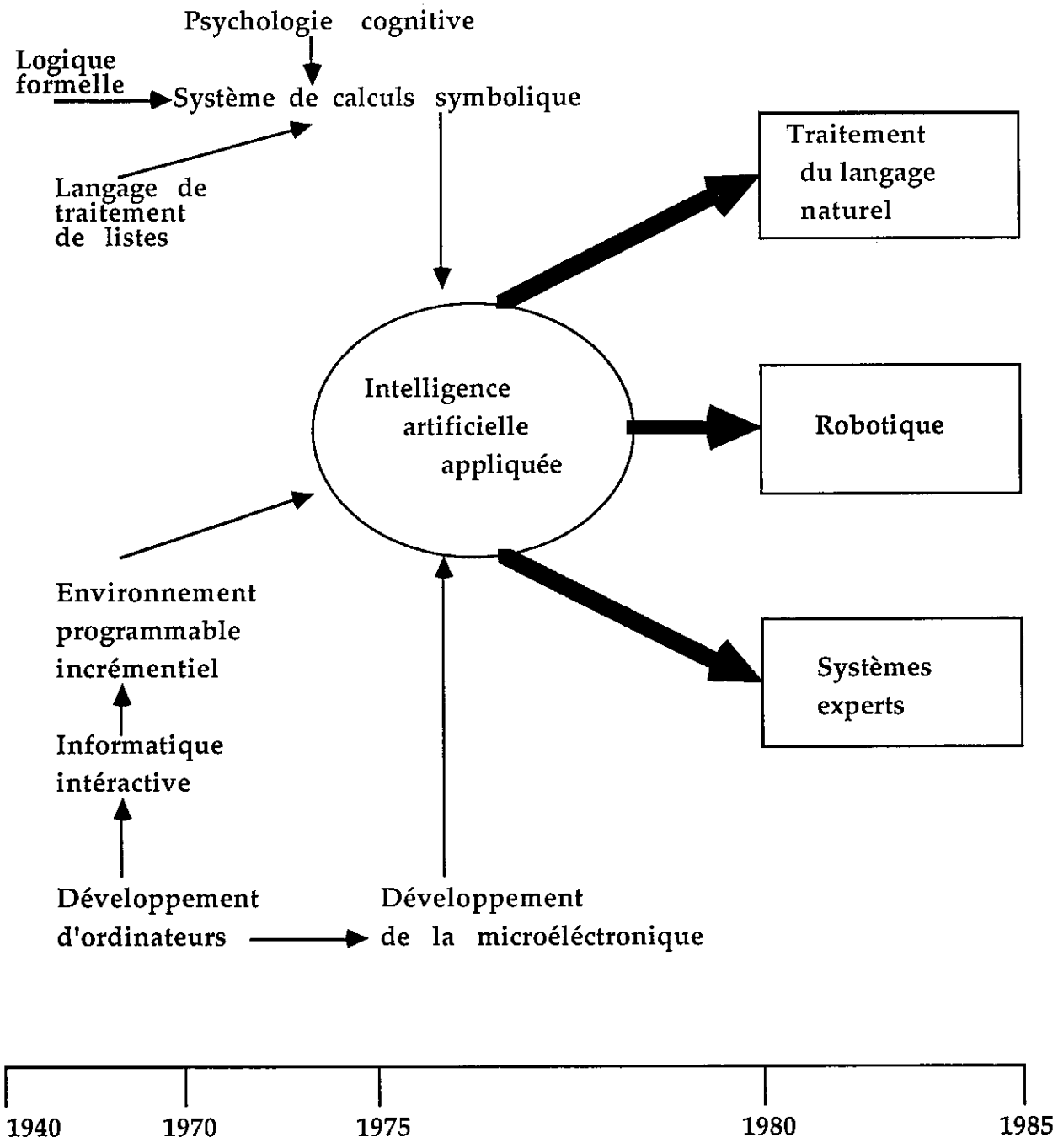
Traitement du langage naturel :
Dans cette partie, les recherches sont orientées vers le développement des programmes informatiques capables de lire , de parler ou de comprendre le langage naturel .

Robotique :
Dans ce domaine, les recherches sont spécialisées au développement des robots intelligents et en particulier des programmes visuels et tactiles permettant aux robots d'assimiler les modifications de son environnement à mesure qu'ils se déplacent .

Système expert :
Dans ce troisième domaine les chercheurs connus sous le nom de cogniticiens sont dirigés vers la conception des programmes qui utilisent les connaissances symboliques pour simuler le raisonnement humain .

---

* Alan TURING est l'un des chercheurs scientifiques possedant une forte connaissances sur la logique formelle



L'évolution des systèmes experts

Machine Universelle (MU) :

On dispose d'un langage de programmation L, et cela nous permet de décrire, d'écrire ou de construire tel algorithme Gamma .Trés généralement cet algorithme est déstiné à s'appliquer sur une configuration Cx elle même appartenant à l'ensemble des configurations C que l'on veut sous mettre à χ

χ : Cx --------> Cy

Algorithme Configuration initiale configuration résultat

On construit une machine universelle qui va travailler sur une configuration C qui aura comme résultat une configuration C' définie comme suit :

MU : C ----------> C'

C : étant le couple

$[R_1(\chi), R_2(Cx)]$ ----------> $[R_1(\chi), R_2(Cy)]$

representation mémoire du code de χ — algorithme — représentation de la configuration sur mémoire

On aura comme résultat :

C' défini par $R_1(\chi), R_2(Cy)$

. L'algorithme χ est stable par la représentation .

. Cy est la configuration résultat de Cx soumis à χ.

Donc la représentation de la configuration Cx est $R_2(Cx)$ transmis à la machine universelle (MU) qui va donner la représentation de Cy notée $R_2(Cy)$.

On peut déduire une définition plus conçise de la machine universelle (MU) :

La MU est un dérouleur possedant deux (02) paramètres qui sont :

. Le code programme (le code du compilateur).

. La configuration du compilateur .

Position du problème en système expert

Génie logiciel et génie cognitif :

Pour résoudre un problème donné , on le définit en terme d'états et d'opérateur, ensuite une génération du domaine de recherche. Le résultat obtenu à partir de cette définition est équivalent à une représentation d'un problème en intelligence artificielle .

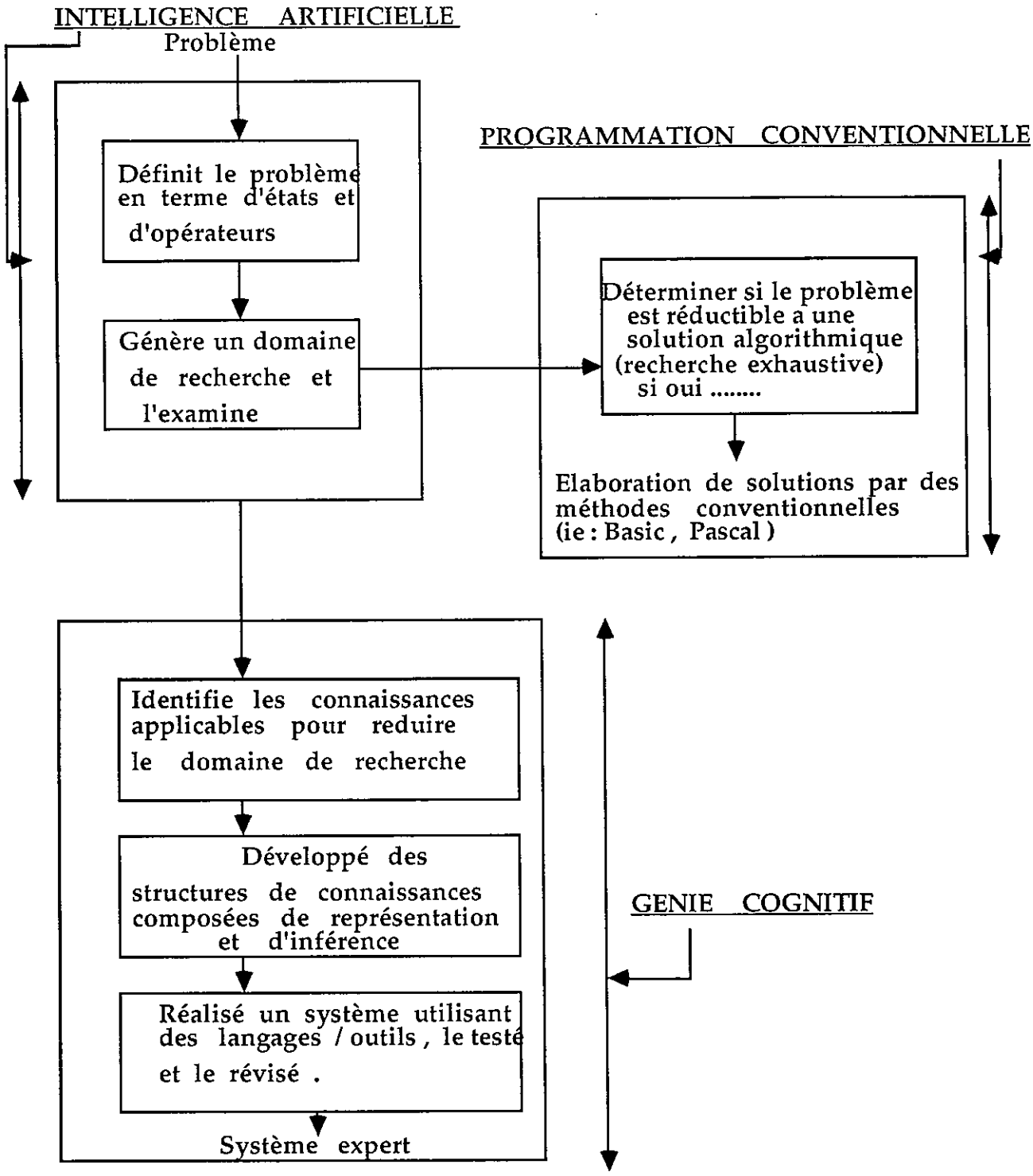
Il existe deux approches différentes pour résoudre un tel problème ,soit par une détermination si le problème donné est réductible à une solution algorithmique (recherche exhaustive) .Et si c'est le cas alors à partir des langages de programmation (ex: pascal,basic ,cobol,...etc) on aboutit à des solutions .Ce genre de résolution de problème s'appelle Résolution par la Programmation Conventionnelle ou plus généralement connu sous le nom de génie de logiciel. L'autre type est celui de la recherche heuristique c'est à dire que l'approche de la résolution est celle de l'identification des connaissances fiables et applicables pour réduire le domaine de recherche .Par la suite un développement des structures des connaissances, composées de représentation et d'inférences qui se conclut par la réalisation d'un système utilisant ces outils , le testé et le révisé pour aboutir à un système expert .On appelle ce genre de résolution de problème résolution par la programmation symbolique connu généralement sous le nom de génie cognitif.

Machine Universelle (MU) :

Résoudre un problème en utilisant les notions fondamentales de l'informatique (Machine Universelle) revient à definir le problème sous forme d'algorithme et de configuration.

Résoudre un problème en génie logiciel et génie cognitif represente une définition du problème sous forme d'opérateurs, d'états et une génération du domaine à étudier.

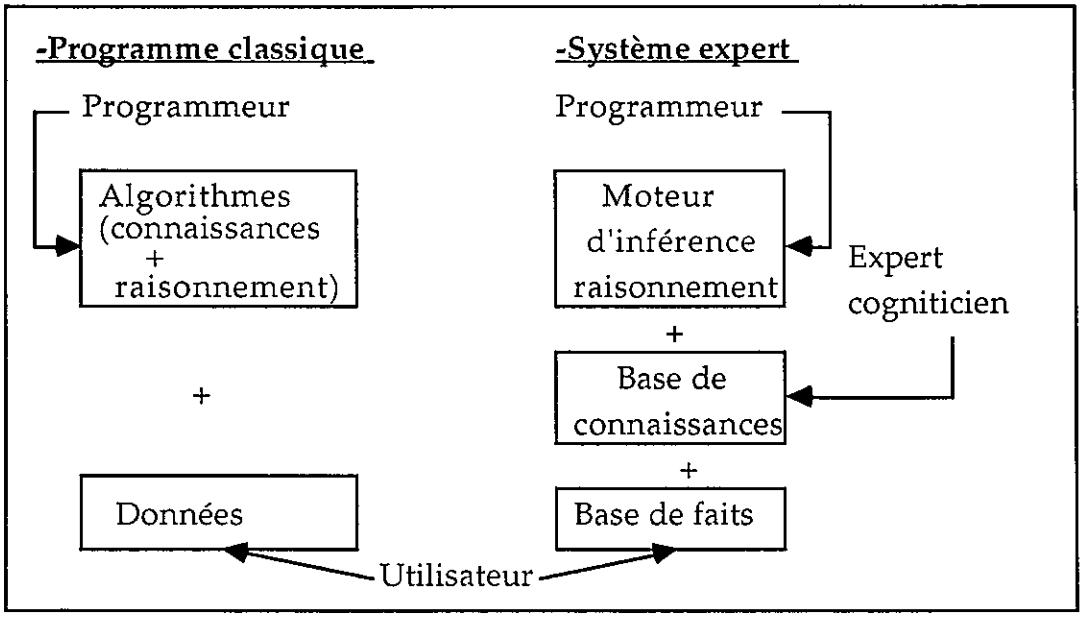
On peut conclure que les opérateurs n'est autres que que des algorithmes et les états sont des configurations .


INTELLIGENCE ARTIFICIELLE
Problème
Définit le problème en terme d'états et d'opérateurs
Génère un domaine de recherche et l'examine
PROGRAMMATION CONVENTIONNELLE
Déterminer si le problème est réductible a une solution algorithmique (recherche exhaustive) si oui .....
Elaboration de solutions par des méthodes conventionnelles (ie : Basic , Pascal)
Identifie les connaissances applicables pour reduire le domaine de recherche
Développé des structures de connaissances composées de représentation et d'inférence
Réalisé un système utilisant des langages / outils , le testé et le révisé .
Système expert
GENIE COGNITIF

Principales différences entre programmation conventionnelle et programmation symbolique :

conventionnelle	symbolique
-Algorithmes	-Heuristiques
-Base de données à adressage numérique	-Base de connaissances structure symbolique au sein d'une base de faits générale .
- Traitement séquentiel	-Traitement fortement interactif
- Démarche procédurale	-Démarche déclarative

Schématisation d'un programme classique et un système expert :



Critiques sur les comparaisons faites sur un programme classique et un système expert :

la comparaison faite entre un programme classique et un système expert n'est autre qu'une comparaison évidente et non scientifique .
En effet :
comparaison évidente:

la démarche procédurale est lourde et peu souple car la modification des informations tendant à devenir très complexe.Par contre la démarche déclarative permet la plupart du temps de faire la mise à jour d'un ou plusieurs éléments de la connaissance sans gêner la bonne marche du processus.Il est utile pour un domaine évolutif de le résoudre par la démarche adaptée au systèmes experts .
Critique :

Cette compariason est évidente car la propriété de faire des mises à jour sans géner la bonne marche du processus est général pour tout système.
comparaison non scientifique :

Moteur d'inférences , Base de connaissances et Base de faits représentent les élément de base d'un système expert , ces derniers sont considérés comme de nouveaux éléments par rapport aux éléments et aux concepts de base de l'informatique fondamentale.
Critique :

Ces éléments ne sont autres que les propriétés fondamentales de la machine universelle (Code programme, Algorithme et la Configuration).

Conclusion de la comparaison Machine Universelle et Système Expert :

Machine universelle (MU):

La Mu est unique , on la construit une fois pour toute et possedant la propriété de simuler le calcul décrit par n'importe quel algorithme travaillant sur une configuration propre à chaque algorithme .
Système expert :

Le moteur d'inférence est unique possedant un mode de raisonnement précis (chainage avant, chainage arrière et chainage mixte) travaillant sur une base de connaissances statiques tout en utilisant des faits sauvegardés d'une base appelé base de faits dynamique.

Types de recherches utilisés

Les types de recherches existantes sont :

La recherche aléatoire :

C'est une anti-méthode qui explore l'arbre au hasard , sans garder trace de son itinéraire . C'est ainsi qu'elle ne va pas toujours jusqu'au bout de la voie parcourue et peut emprunter plusieurs fois le même passage. Une recherche aléatoire peut parfois atteindre rapidement sont but(le cercle) par hasard mais c'est la façon la moins efficace d'y parvenir .

La recherche exhaustive :

En explorant systématiquement l'arbre selon un plan prédéterminer , cette approche permet de tester chaque voie une fois , sans en oublier ni utiliser deux fois la même . Cette forme de recherche part toujours par exemple d'une voie de gauche, à moins qu'elle n'ait déjà été explorée , et bat en retraite quand une voie à été épuisée, jusqu'au but à atteindre (le cercle).

La recherche heuristique :

A tout carrefour de décision , elle applique une heuristique de choix, règle d'évaluation estimant les chances de chaque option de conduire au but , puis suit la plus prometteuse rebroussant chemin si nécessaire. Ici , l'heuristique consiste à mesurer la distance entre un point et le but (le cercle) , puis à suivre le chemin qui permet de s'en approcher le plus près possible .

Système expert :

Définition d'un système expert :

Un système expert peut être considèré comme un ensemble de Connaissance, Raisonnement et Interfaces destiné à résoudre un problème particulier.

Définition formelle d'un système expert :

Un système expert est composé de trois parties principales (Base des faits,Base de connaissance ,Moteur d'inférence) qui utilisent les même éléments de connaissances ' faits ' .

Base de faits :

La collection des informations touchant un utilisateur , qui représente une mémoire temporaire pour le système expert .Elle est mise à jour par la progression du raisonnement est vidée lorsque une consultation se termine .

Base de connaissances :

Elle rassemble la connaissance et le savoir faire d'un expert .Il existe deux types de connaissances .

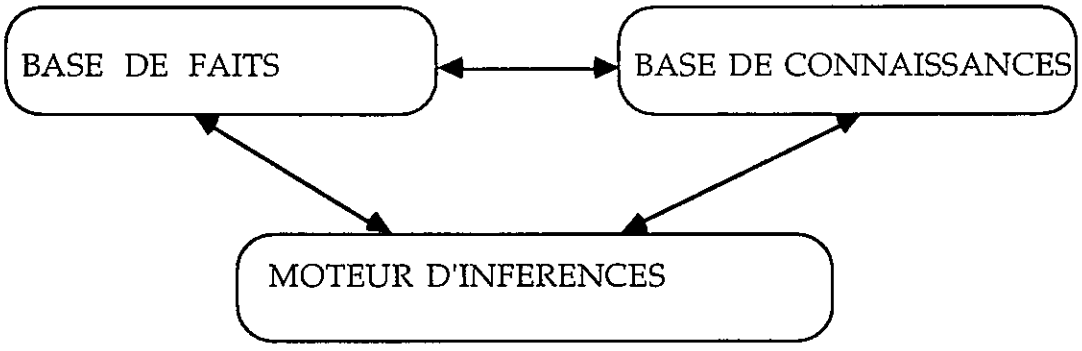
- des connaissances indiscutable .
- des connaissances heuristiques ou pragmatiques c'est à dire un ensemble de connaissances intuitives regroupant les convictions personnelles de l'expert .

Moteur d'inférences :

C'est un mode de raisonnement regroupant des mécanismes de déduction et un programme général d'interprétation des connaissances .Il suffit de lui introduire une base de connaissances d'un domaine particulier pour qu'il simule le raisonnement d'un expert humain .

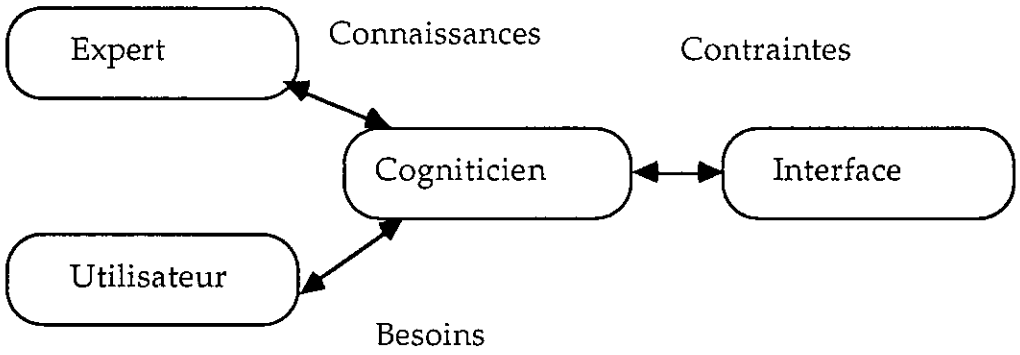
Architecture d'un système expert :

Architecture simple



Architecture avec interface

Dans cette architecture est utilisée Une interface linguistique particulière pour faciliter le dialogue entre l'utilisateur et le système . Ce module d'interface contient des programmes de compréhension du langage naturel ,de correction de fautes d'orthographe et de mise en forme des réponses et d'explications ainsi que les questions posées par le système expert .

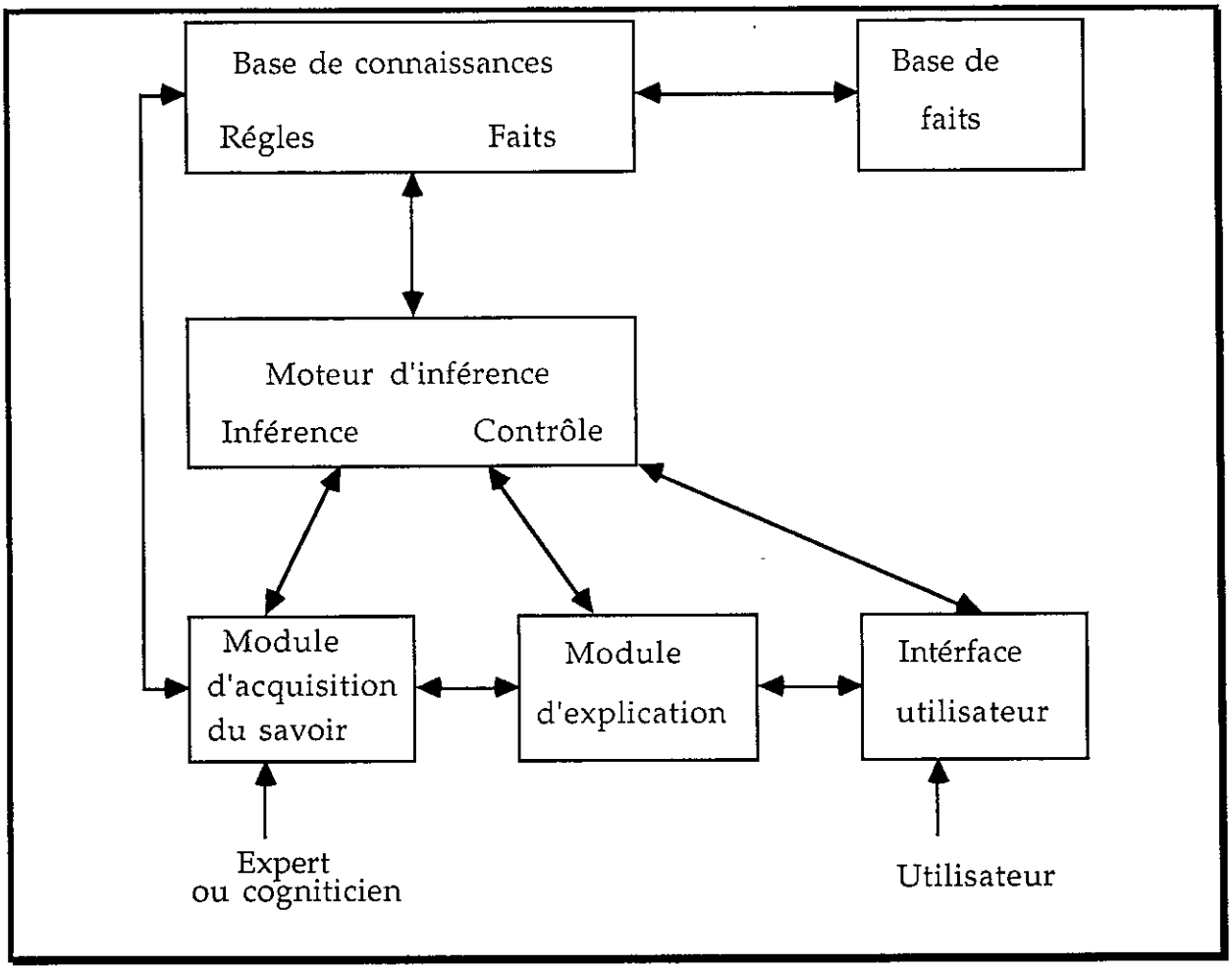


Remarque :

La présence de ce module est pratiquement indispensable car il rend le système accessible à l'utilisateur ou a l'expert,pour le premier pouvoir dialoguer avec le système dans un langage souple et pour le second une facilité de modeler et de modifier les connaissances sans faire recours a une étude approfondie des systèmes experts ou aux langages de programmation .

Architecture complète

Cette architecture prend en compte l'intégrité du système expert par rapport au système d'information d'un environnement donné .C'est à dire la communication ou plus exactement l'échange d'informations dans les deux (02) sens entre système expert et son environnement .



Personnages participant à la réalisation d'un Système Expert :

les personnages sont l'expert,l'utilisateur et le cogniticien chacun d'eux joue un rôle déterminant pour la réalisation d'un Système Expert .

Rôle du cogniticien (Ingénieur de l'information):

Les Cogniticiens aident activement les experts à formaliser leur mode de résolution des problèmes .

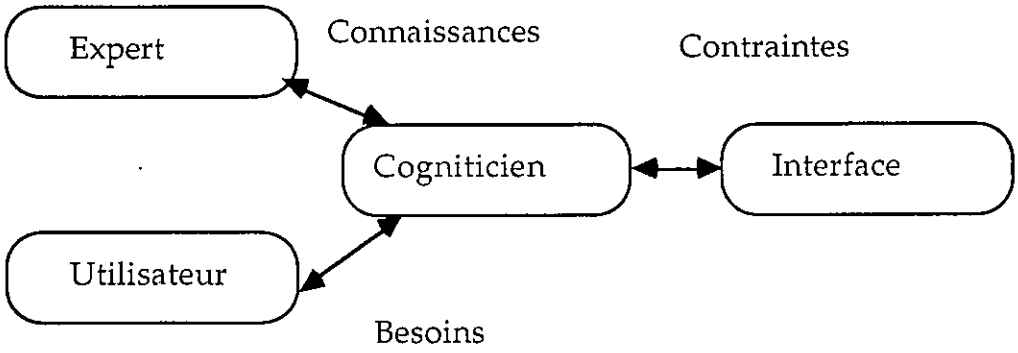
Rôle de l'expert:

Un Expert est une personne largement reconnue comme pouvant résoudre un type particulier de problème que la plupart des autres personnes ne peuvent effectivement pas le résoudre aussi efficacement.

Utilisateur :

La personne qui utilise le système après avoir été conçu.

Processus de transfert des informations



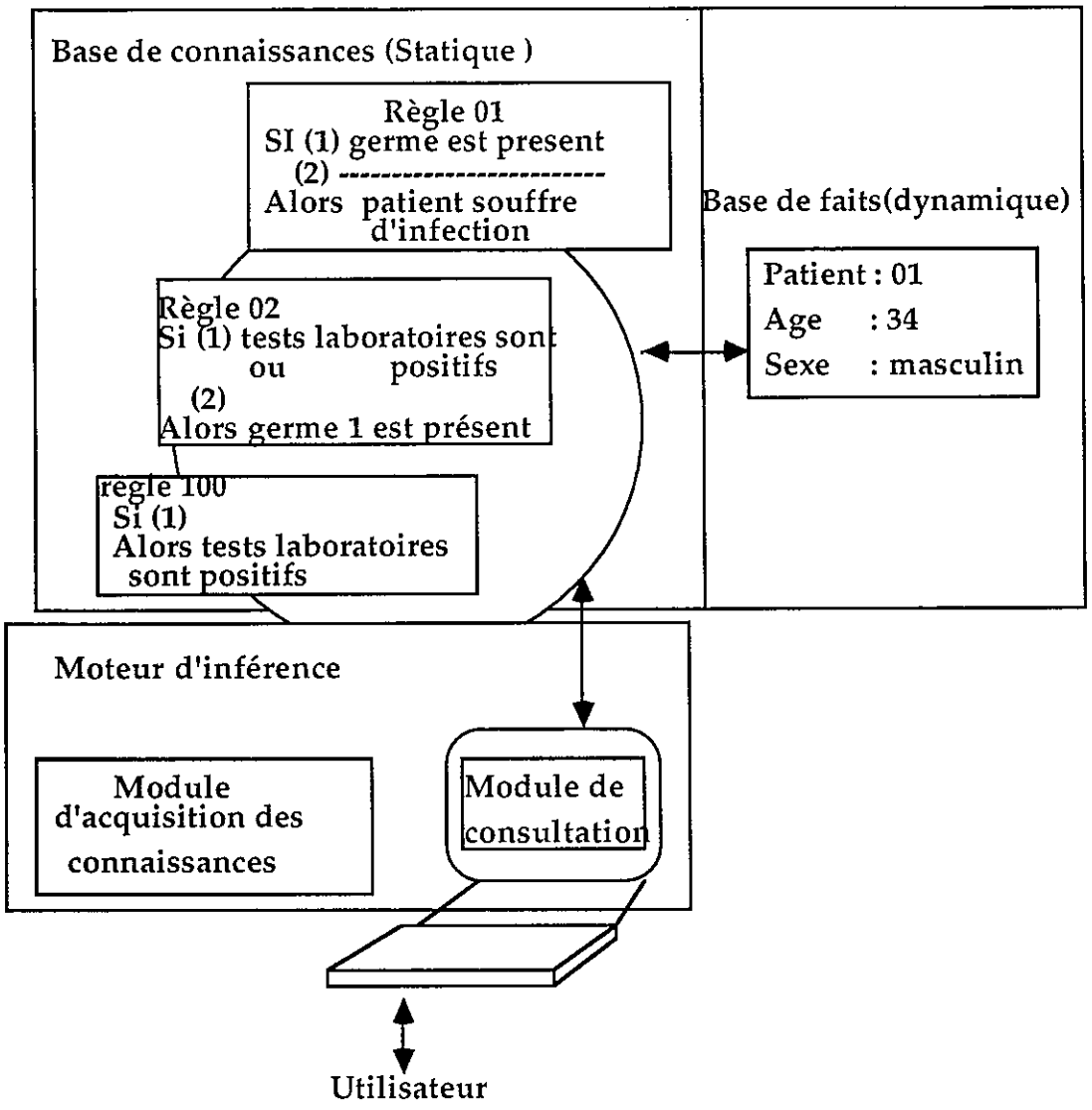
Stratégies de représentation des connaissances

Pour modéliser la connaissance d'un expert ,il existe plusieurs stratégies de représentation. Actuellement la grande majorité des systèmes experts (plus de 90%) utilisent les règles de production comme moyen de représentation des connaissances .Ces règles de production sont aussi utilisées par exemple dans les algorithmes de MARKOV puis dans les règles de réécriture de CHOMSKY pour décrire la grammaire .Une règle de production est de la forme **SI antécédents ALORS conclusion** qui est équivalente à **SI condition ALORS action** .Le couple (antécédent,conclusion) ou bien (condition ,action) représente des faits qui peuvent être représenté de trois (03) manières différentes :

- Par la logique des propositions pures qui se formalise par un nom symbolique (exemple: douleur ,maison,...etc).
-Par la logique des propositions qui se formalise par un quadruplé <Objet Attribut Comparateur Valeur> (exemple :Température intense = élevée).
-Par la logique des prédicats qui se formalise par un couple <Prédicat Variable > (exemple : Pers(?x ?y)).

Exemple de représentation d'un système expert:

L'exemple est tiré d'un système expert médical , on peut avoir plusieurs passions pour une seule base de connaissances médicale ,et les informations fournies par ses passions représentent des bases de faits .D'ou on peut tirer la conséquence suivante : La base de connaissances est statique(par rapport a la base de faits mais elle est dynamique par rapport aux nouvelles dècouvrertes) par contre la base de faits est dynamique .



Caractéristiques d'un système expert :

Les systèmes experts se différencient principalement de point de vue externe par le langage qu'il utilise (c'est à dire la puissance d'expression et la formalisation des situations complexes) qui dépend éventuellement de la qualité du langage de représentation des connaissances , et de point de vue interne par la démarche déductive qu'ils proposent . En d'autres termes la puissance d'un système expert dépend de la qualité des informations qu'il posséde(c'est à dire son expertise) et de la qualité de ces déductions .

Conséquences

Il est clair que le langage et le moteur d'inférence sont inter-indépendant.

Fonctionnement d'un système expert

Le fonctionnement d'un système expert veut dire le fonctionnement du Moteur d'inférence. Ce fonctionnement se traduit par une intérpretation des connaissances jusqu'à la satisfaction des conditions d'arrêt ,en passant par trois phases essentielles qui sont :

- Détection des règles intéressantes .
- Sélection de la règle à appliquer .
- Déclenchement de la règle retenue .

L'intérpretation des connaissances possède deux modes de raisonnement, le premier est celui du raisonnement déductif (chaînage avant), et le second est celui du raisonnement inductif (chaînage arrière) , le choix de l'un ou l'autre raisonnement dépend de la nature du problème à résoudre .

Définition du raisonnement déductif

On utilise le chaînage avant lorsque l'on a pas d'idée précise sur l'objectif à atteindre .Cette démarche est la suivante : partant d'un ensemble de faits le système sature sa base de connaissance ,il s'arrête lorsque plus rien ne peut être déduit. Comme exemple de système expert avec chaînage avant : un juriste partant d'un ensemble de faits , cherche une solution juridique.

Définition du raisonnement inductif

Dans ce cas le moteur d'inférence cherche à atteindre un ou plusieurs buts, c'est à dire que le but à atteindre est défini. Le moteur d'inférence essaye d'appliquer les règles qui concluent sur ces buts.

Il existe aussi un autre mode d'intérpretation qu'on appelle chaînage mixte qui est un mélange des deux modes d'intérpretation .

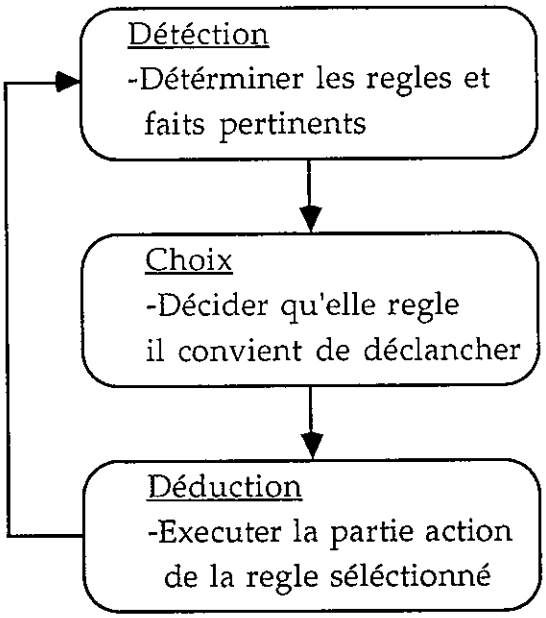
fonctionnement du chaînage mixte : << Le raisonnement déductif(chaînage avant) peut appeler le raisonnement inductif (chaînage arrière) >>,lorsque certains faits n'ont pu être déduis , et réciproquement << Le raisonnement inductif peut appeler le raisonnement déductif >> lors d'une résolution .

Remarque :

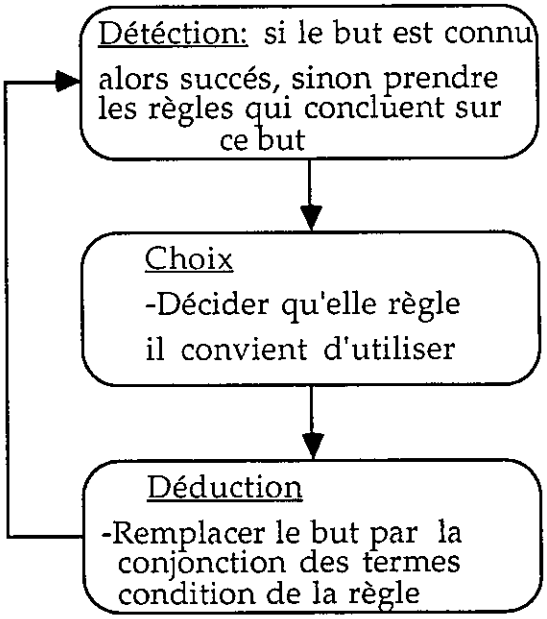
Le deuxième cas est très fréquent et permet lors de la vérification d'un but de déduire les informations supplémentaires.

Présentation des cycles de base :

Cycle de base d'un intérprete déductif (les différentes phases)



Cycle de base d'un intérprete Inductif (les différentes phases)



La vie d'un système expert

La vie d'un système expert passe par trois phases essentielles :

PHASE N°01 : Développement

Dévéloppement d'un petit système expert opérationnel

PHASE N°02 : Extension

Apartir d'un système expert opérationnel original on ajoute des connaissances et des fonctionnalités.

PHASE N°03 : Restructuration

A cause de l'évolution téchnologique ou de limitation des systèmes experts trop contraignantes , il peut être nécéssaire de transformer radicalement le système expert initial.C'est à dire la mort d'un SE suivi d'une réincarnation

Développement d'un système expert

Pour une bonne maîtrise du développement d'un système expert , connaître la méthode de conduit d'un projet informatique en utilisant les approches universelles (exemple : l'approche Merise) serait souhaitable . l'adoption ce genre d'approche pour le développement des systèmes experts donnera aux concepteurs (Cogniticiens , Informaticiens) la possibilité de traiter tous les cas possible du domaine à traiter .

Le développement d'une application informatique passe par trois étapes fondamentales :

Etape N° 01 : Etude préalable

-Description avec fidélité la situation existante d'une activité donnée.

-Détermination des besoins ou des objectifs .

-Critiques et suggestions de la situation existante de l'activité .

Etape N° 02 : Etude conceptuelle

-Dégagement des différents traitements après effacement de l'organisation de l'activité.

-Comparaison entre les traitements théoriques et les traitements existants pour déterminer l'écart.

-Regroupement et structuration des données .

A partir des nouveaux besoins , des traitements dégagés , écart de traitement , on déterminera les nouveaux traitements qui seront présentés sous formes d'un ou plusieurs processus .Puis , en intégrant l'organisation et l'outil informatique chaque processus se transforme en une ou plusieurs procédure .

Etape N° 03 : Réalisation

La dernière étape consiste à déterminer les différents programme et de les implanter sur un matériel informatique choisi ou destiné.

Par contre au niveau système expert l'étude ou l'analyse préalable consiste à choisir un problème qui se traite efficacement par les systèmes experts ,ensuite fixer les objectifs à atteindre .

Au niveau de cette étape on distingue deux parties complémentaires ; la modélisation (c.a.d l'acquisition des connaissances de l'expert au cogniticien) ,et son analyse fonctionnelle (c.a.d l'analyse par le cogniticien de ces connaissances pour qu'elles soient prête à être fournie a la machine .

Extension d'un système Expert

Extension Système expert = Système expert + Outils de développement du moteur d'inférence sans connaissances (**SHELL*****)

Remarque :

L'acquéreur de cette coquille devra y introduire lui même de la connaissance. C'est équivalent à un compilateur ,il reste à l'acheteur de ce compilateur à écrire son programme .

Généralement les cogniticiens développent les systèmes experts par domaine d'activité : médecine , agriculture , chimie , droit , électronique /informatique , Environnement /géologie , militaire /spatiale .

Les systèmes experts peuvent être aussi développés par fonction d'entreprise: vente, production , conception d'un produit , gestion . Développement des systèmes experts par fonction Gestion : analyse financière , analyse des risques , gestion de la trésorerie , constitution automatique des documents , dépouillement des questionnaires, recherche d'aides publiques , aide achat , gestion des ressources humaines , choix d'investissements , aide au marketing . D'autres systèmes experts peuvent être développés par types comme : diagnostic, prévision, interprétation,Surveillances, commandes de système réactor,Conception ,planification , enseignement .

*** terme anglais qui veut dire coquille vide

Problèmes des systèmes experts

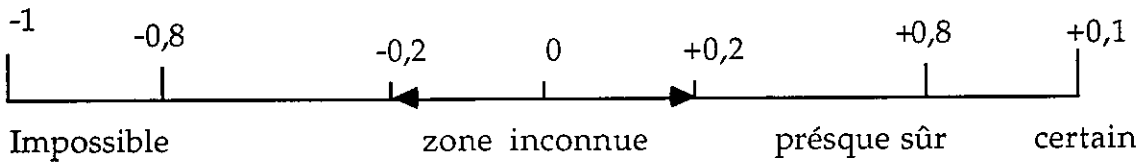
Formalisme de représentation

Une des taches délicates lors de la conception d'un système expert est la détermination d'un formalisme adéquat pour représenter ces unités de connaissances.

L'expression de l'incertain

Il existe plusieurs méthodes pour manipuler l'incertain ou le douteux qui sont les méthodes probabilistes, les coefficients de vraisemblance, la théorie des ensembles flous, les logiques modales ou multivaluées.Tous cela dans le cadre de la logique classique vrai ou faux .

Il est possible qu'un fait ne soit pas complètement évident alors dans ce cas l'objet-attribut-valeur est modifié par un coefficient appelé facteur de vraisemblance pour refléter la confiance que l'on a dans un fait. Le facteur de vraisemblance peut être représenté de manière très diverses. En prenant comme exemple le système expert MYCIN les nombres utilisés varient de -1 à +1; le -1 représente un fait complètement faux et son opposé représente un fait complètement vrai.



Le but de son utilisation :

Par comparaison au raisonnement de l'expert humain ,les experts humain ont souvent l'habitude de prendre des décisions à partir des informations incomplète.

Intérpretation de l'incertain:

Toujours en prenant l'exemple du système expert MYCIN, les degrés de certitude sont gérés de trois manières :

1- Les faits peuvent être obtenus à partir de plus d'une régle c'est à dire une fonction de composition mixe les facteurs de vraisemblance.

2- Les premisses composées(celles qui a plusieurs clauses reliées par les opérateurs ET ou OU) estiment les faits incertains .Donc une premisse incertaine conduit à une conclusion incertaine .

3- Soit l'exemple suivant :

R1 ----> SI température=intense(0,6) ALORS action1
R2-----> SI objet2 = intense(0,5) ALORS action2

Le moteur d'inférence compose les deux valeurs , ainsi en les composants on trouve la certitude sur la première conclusion à augmenter de 0,2 .

Maintenance d'un système expert

Un système expert à besoin d'une maintenance comme un programme informatique. C'est à dire il ne suffit pas de livrer un système expert et de ne plus s'en occuper; car livrer un système expert en désignant l'expert pour faire des mise à jour ou des compléments de connaissances peut être très dangereux si l'expert n'as pas les compétences informatiques .

Terminologie utilisée :

Algorithme :Procédure systématique qui,si elle est suivie ,garantit un résultat correct .

Bloc :Ensemble de faits stockés et rappelés en un seul morceau.

Fait : En gros ,assertion dont la validité est reconnue .Dans la plus part des systèmes des experts ,un fait est constitué d'un attribut et d'une valeur spécifique associé .

Inférence :Processus par lequel de nouveaux faits sont déduits de faits connus .

Interface :Lien entre le programme informatique et le monde extérieur .Un seul programme peut disposer de plusieurs interfaces. Les systèmes experts possèdent typiquement pour leur développement(interface d'acquisition du savoir)et pour les utilisateurs (interface utilisateur). De plus certains systèmes disposent d'interfaces pour échanger de l'information avec d'autres programmes ,avec des bases de données ,des organes de visualisation ou de capteurs .

Procédure conventionnelle et procédure déclarative :Deux perspectives complémentaires d'un programme informatique.La procédure dit au système se qu'il doit faire ;la déclaration dit au système ce qu'il doit savoir .

Recherche exhaustive :Une recherche est exhaustive si tous les chemins d'un arbre de décision ou d'un réseau ont été explorés.

Système de production: C'est un système informatique ou humain qui dispose d'une base de règles de production et de quelques mécanismes de contrôle qui choisissent les règles à appliquer afin d'atteindre un état but .

Vraisemblance: Degré de confiance que l'on a dans un fait ou une relation . Utilisé en intelligence artificielle, il s'oppose aux probabilités qui mesurent la possibilité qu'un événement se produise .

REFERENCES BIBLIOGRAPHIQUES

[LAURIERE88] J.L.LAURIERE
Intelligence Artificielle Tome II :
Représentation des connaissances
Ed : EYROLLES

[LAURIERE87] J.L.LAURIERE
Intelligence Artificielle :
Résolution des problèmes par l'homme est la machine
Ed : EYROLLES

[DELAHAYE87] J.P.DELAHAYE
Organisation et programmation des bases de connaissances
Ed : EYROLLES

[DELAHAYE87] J.P.DELAHAYE
Outils logiques pour l'Intelligence Artificielle
Ed : EYROLLES

[BOUDIBA89] E.BOUDIBA
Amélioration de l'algorithme de production
Bulletin d'Informatique approfondie et applications
ISSN 0291-5413 N° 26,27 Septembre et Décembre 1990

[BOUDIBA88] E.BOUDIBA
Conception d'un système d'information automatisé
pour le suivi de la maintenance
Mémoire d'Ingénieur en Informatique
à I.N.I (ex- CERI) Alger Algerie

[VOYER87] R.VOYER
Moteur des systèmes experts
Ed: EYROLLES

[PITRAT85] J.PITRAT
Textes, Ordinateurs et compréhension
Ed : EYROLLES

[BRIAND88] R.BRIAND
Méthodes de développement des systèmes experts
Ed : EYROLLES

[HART88] A.HART
Aquisition du savoir por les systèmes experts
Ed : MASSON

[TSI89] Numéro spécial systèmes experts
Technique et science informatique
Ed : DUNOD AFCET

[BIANCO76] E.BIANCO
Informatique fondamentale :
De la machine de Turing aux ordinateurs modèrnes

UOUZZAUEDIBISAR

Les aventures du Vicompte Jacques-Etienne Xavier de L'Or.

Fort impressionné depuis l'âge le plus tendre, par un nom qui lui pesait lourdement sur les épaules, et de l'ombre duquel il essayait de se dégager, le Vicompte se voulait de devenir un grand homme. Fi d'une Destinée Nationale ... Trop mesquin! L'Europe se profilait à l'horizon, il se saisit de sa chance, et il enfourcha ce ô combien plus noble Destrier.

C'est d'ailleurs vers la même époque qu'il ressentit à quel point son Destin était Social.

Avec la volonté d'abord, puis le talent, puis le génie, la compétence, puis la renommée, et enfin la gloire surgirent. C'est ainsi que le Vicompte Jacques-Etienne Xavier de L'Or voyait le Personnage qu'il admirait chaque matin, dans sa glace, au saut du lit, tout nu, à l'instant incertain où il faut un peu se rassurer.

Un beau matin, il eut pitié de cette image qui lui faisait face, de ce corps rompu à toutes les tâches ingrates de l'Administration. Les paupières laissaient filtrer un regard rétréci visiblement habitué à réduire le champ de vision à l'essentiel, les chiffres. Les côtes saillaient sous une peau blanchâtre, au dessus des deux pointes d'os du bassin, de part et d'autre de la petite virgule rose et molle, inutile, dans sa petite touffe de cresson qui commençait à grisonner et à se clairsemer. Enfin les jambes comme des fils de fer barbelés avec des pointes partout, les genoux, les chevilles, les orteils ... Non qu'il eût honte de son anatomie, depuis belle lurette ce n'était plus pour lui un moyen de vaincre, mais il sentait que sa carcasse avait besoin, sinon de repos, du moins de montrer qu'on avait gagné, et que le temps de prendre du recul était venu. L'expression lui vint à l'esprit: "se reposer quelque peu sur ses lauriers". Il s'imaginait fort bien une tête de César, portée dignement, avec la couronne de lauriers sur le front.

Jacques-Etienne Xavier décida donc de s'accorder quelques jours de vacances, comme ça, tout simplement.

Mais tout n'est pas si simple.

Où aller?

Qu'y faire?

Le vicompte convoqua son Chef de Cabinet et lui tint ce langage: « Mon cher Oscar, j'ai décidé de m'abandonner quelques instants aux charmes de la nature ... Trouvez moi l'endroit idéâl ! ».

Et le cher Oscar s'en fut.

Il jeta son dévolu sur la région de Zwuidcoote. Les plages du sud? trop chaudes et trop polluées de vaine populace. Les plages de l'ouest? trop de risques d'un naufrage de ces gros bidons flottants bourrés

d'horrible mazout. Les plages d'Italie? le peuple Italien, certes artiste, est si vainement bavard et agité.

Subjugué par une telle aubaine, Monsieur le Maire de cette charmante bourgade s'empressa de guider Oscar vers la plus magnifique de ses plages, celle qui faisait son orgueil.

« Qu'en pensez-vous de ce magnifique sable argenté ...» dit le Maire d'un geste large et d'une voix sonore, en bombant le torse, comme un tribun qui sait comment haranguer la foule.

« Très très beau, en effet ... » dit Oscar, mais il ajouta à mi-voix: « ... cependant voyez vous, Monsieur le Vicompte adore le sable doré d'Italie, surtout quand il est parsemé de délicates tâches rouges de sable d'Espagne ...»

On s'empressa de faire venir cent soixante et douze trains de sable d'Italie, puis trente et un trains de sable d'Espagne. Et Monsieur le Maire reçut à nouveau Oscar qui se déclara tout-à-fait satisfait pour le sable désormais magnifiquement doré avec des petites plages rouges du plus bel effet réalisées par les plus grands artistes.

« Et admirez cette mer d'un délicat bleu-gris ...» ajouta-t-il sur le ton de l'orateur qui sonde les foules.

« C'est vrai, c'est vrai ...» dit Oscar, « mais voyez-vous Monsieur le Vicompte adore la mer bleutée avec de belles traînées vertes et mouvantes ...» .

Qu'à cela ne tienne, par l'intermédiaire du Préfet et des affaires étrangères, on fit activer deux sous-marins de la Royale, et deux submersibles jaunes de sa Gracieuse Majesté, afin de répandre au bon moment, quelques belles traînées de fluorescéine à quelques encâblures du rivage.

Quand Oscar fut mis au courant, il félicita chaleureusement Monsieur le Maire, qui lui fit alors, avec quelque appréhension toutefois, admirer son ciel fantastiquement chargé de puissants nuages.

« Aaaa ...» répondit Oscar « ce sont de merveilleux nuages, mais vous savez, Monsieur le Vicompte adore les cieux très bleus, très dégagés avec juste quelques petits cumulus de beau temps ...»

« Ah ... » fit le Maire en avalant sa salive « Ah ...»

Et l'on fit une enquête, on lança des Sociétés d'Etudes sur l'affaire, des politiciens de renom y furent mêlés, et l'on aboutit enfin à la conclusion qu'il fallait faire appel à l'US Air-Force. Et il fut décidé de faire croiser douze cent quarante deux bombardiers, parmi les plus lourds, bourrés de neige carbonique pour faire crever les nuages, la veille de l'arrivée du Vicompte, à la suite de quoi quelques Mirages suffiraient pour déposer quelques petits cumulus artistiques çà et là.

Et vint le grand jour.

Le Vicompte Jacques-Etienne Xavier de L'Or, enfin en vacances, accompagné de son fidèle Oscar débarque sur la plage de Zwuidcoote. Il

foule aux pieds ce merveilleux sable doré délicatement moucheté de rouge, dans l'eau, les reflets améthystes sur un fond de gris-bleu lui réchauffent le cœur, dans le ciel d'un azur éclatant, de délicieux cotons nuagers lui ravissent les yeux...
« Ah que la vraie Nature est belle ...» soupire le Vicompte.

Adaptation par
J. W. W. H. Woolworth