ORSAY, Septembre 1988

Rapport de stage de D.E.S.S. présenté par : M^elle Agnès ETIFIER

APPLICATION DES SYSTEMES EXPERTS A LA CARTOGRAPHIE PAR TELEDETECTION

REMERCIEMENTS

Je ne saurais commencer ces remerciements sans m'adresser en premier lieu à mon professeur Monsieur J.G. Ganascia qui m'a proposé ce stage, m'a dirigée lors de mes recherches, et qui par ses conseils, m'a aidé à mettre en forme ce rapport.

Je tiens à remercier tout particulièrement Madame Catherine Mering qui m'a encadrée pendant ce stage avec beaucoup de gentillesse, et qui a été pour moi d'une aide et d'un soutien sans lesquels il m'aurait été difficile d'effectuer correctement ce stage.

Il m'aurait été plus difficile de comprendre l'ensemble du projet sans l'intervention de Monsieur François Monjanel qui, par ses explications, ses conseils et sa disponibilité, m'a permis de mieux cerner les problèmes qui se posaient à moi, et d'en éviter d'autres. Pour sa gentillesse aussi je voudrait le remercier ici.

Je voudrais terminer ces remerciements en rendant hommage à toute l'équipe du L.I.A. qui est formidable de gentillesse et de sympathie à mon égard, et qui m'a accueillie sans réserve.

INTRODUCTION

Ce rapport rend compte des premiers travaux que j'ai effectué dans le cadre de mon stage à l'ORSTOM (Institut Français de Recherches Scientifiques pour le Développement en Coopération), au Laboratoire d'Informatique Appliquée. Mes travaux au sein du L.I.A. ne sont pas terminés : l'analyse que je présente ici sera suivie d'une implémentation dont la réalisation à déjà commencé.

LE SUJET DU STAGE

Le L.I.A. est spécialisé dans la recherche en télédétection. En collaboration avec Monsieur Blamont du C.N.R.S., des recherches ont commencé pour étudier à l'aide d'un système expert : CIME, la cartographie des formes végétales dans les régions montagneuses du Nepal. Un système expert a été mis en place en 1987, à l'occasion du stage de D.E.S.S. de François Monjanel [Monjanel87]. Le travail qui m'était demandé était de résoudre les problèmes qui s'ubsistaient, et d'étendre ce premier système.

L'INSTALLATION DU MATERIEL INFORMATIQUE

Le L.I.A. est équipé de deux stations SUN 310 et de 6 terminaux reliés par un réseau interne, avec un accès au réseau TRANSPAC. Par ailleurs, des micro-ordinateurs sont aussi accessibles aux chercheurs (Macintoch.SE (APPLE) et Goupils.G4 et G5)

Nota : -Dans le rapport, tous les mots suivis d'une étoile sont expliqués dans un lexique se trouvant à la fin du document. Pour plus de précisions, je vous invite à vous y reporter.

-Il existe un index se trouvant lui aussi à la fin de ce document.

CHAPITRE I

PRESENTATION : DE L'ANCIENNE VERSION DE CIME A LA NOUVELLE

A] POURQUOI CIME ?

1 Présentation du domaine d'application

Il s'agit de produire des **cartes thématiques*** à l'aide de la télédétection. Dans le cas de CIME, on cherche à produire une carte de la végétation en région montagneuse (Nepal central), à l'aide des données du satellite LANDSAT.

1.1 Cartographie thématique

La cartographie **thématique*** par télédétection requiert une connaissance approfondie du milieu géographique que l'on cartographie, et une maîtrise des traitements spécifiques des données satelitaires. La production d'une telle carte nécessite donc la conjonction de deux expertises : l'une dans le domaine de la géographie (se rapportant plus précisément la région étudiée, et le **thème*** que l'on cartographie), l'autre dans le domaine de la télédétection.

La carte est une représentation conventionnelle plane des phénomènes localisés à la surface terrestre. Une carte est définie par une échelle et une légende (liste de symboles exprimant visuellement l'identité des phénomènes que l'on veut représenter). On note qu'il existe deux types de cartes :

La carte de synthèse : c'est un document permettant de présenter des connaissances. Aucune information nouvelle n'est produite lors de son élaboration : il ne s'agit que d'un instrument d'exposition et de communication

La carte d'analyse est au contraire prospective. Son élaboration exige une analyse de documents (photographies aériennes, images satellitaires). Son but est d'identifier des objets à l'aide d'une technique d'analyse, de la thématique (objectif souhaité), et de connaissances préexistantes sur le milieu à analyser. Une telle cartographie est un moyen de produire des connaissances, et en particulier l'identification et la caractérisation d'un thème dans l'espace. Dans le projet CIME, il s'agit de produire des cartes d'analyse.

1.2 Utilisation de la télédétection pour la cartographie

Ici, le terme **télédétection** est utilisé pour désigner les méthodes d'acquisition de l'information à distance qui utilise l'intermédiaire du rayonnement électromagnétique. Ce sont les caractéristiques propres du rayonnement réfléchi par les objets à la surface de la terre, qui doivent permettre de les distinguer à distance.

Le problème de la télédétection, c'est d'effectuer le passage d'une information radiométrique ponctuelle à une image interprétée, puis à une carte thématique. Les traitements en télédétection ont pour but de résoudre ce problème en réalisant ce passage, sachant qu'à une image donnée peuvent correspondre plusieurs cartes dépendant chacune du thème adopté.

1.3 La cartographie thématique dans CIME

Il s'agit de la cartographie des formes de végétation dans le centre du Nepal. Le travail de terrain a permis de dresser la liste des formations végétales composant le paysage de la région dont on traite l'image. Il a permis plus particulièrement de repérer des parcelles types qui permettront de contrôler la carte. Par ailleurs, la télédétection fournit des informations radiométriques sur ces formations.

La spécificité du paysage étudié se caractérise par :

* Une topographie particulièrement accidentée (forte amplitude altitudinale, pentes abruptes);
* Des expositions diverses des versants montagneux (versants éclairés à l'est, et versants dans l'ombre à l'ouest);
* La petite taille et la grande variété des unités de paysage.

Compte tenu de ces spécificités, les seules données radiométriques ne permettront pas de délimiter les formations végétales.

Pour parvenir à une cartographie, il faut adjoindre d'autres descripteurs du pixel tels que :

* Les indices de texture locaux (permettant de décrire le degré d'hétérogénéité locale);
* Des indices topographiques (l'altitude et la pente)
* Un indice d'ensoleillement (quantité de lumière reçue au moment de l'enregistrement de l'image).

De plus, des connaissances concernant l'organisation du paysage montagnard (étagement de la végétation, pentes maximales permettant l'activité humaine, densité végétale), seront exploitées par le thématicien dans le processus cartographique.

2 Pourquoi un système expert ?

CIME (Cartographie Intelligente en Milieu montagnard), est un système expert destiné au traitement d'images satellitaires. Initialement, les régions étudiées étaient des zones montagneuses du Nepal, le but étant de trouver la meilleure chaîne de traitements pour produire des cartes de la végétation dans une **démarche supervisée** (voir B] § 2).

Avant la mise en place du premier système expert CIME [Avignon88], des démarches procédurales avaient été utilisées. Il s'est alors révélé que ces démarches n'étaient pas satisfaisantes car il n'existe pas de procédure autonome résolvant l'ensemble des problèmes qui se posent en télédétection, où il faut prendre en compte les domaines d'applications, et donc pouvoir piloter les traitements numériques par un raisonnement expert. En effet, dans ces démarches :

- Il n'y a pas de possibilité de reproduction de la méthode qui fait intervenir à la fois les traitements numériques et le raisonnement de l'expert.

- **l'intervention humaine** est toujours nécessaire au cours des traitements, mais ces manipulations de l'expert ne sont pas explicitement représentées dans la démarche. En effet, l'expert part d'hypothèses, effectue une **taxonomie*** au départ, puis a recours à des procédures numériques pour élaborer la taxonomie à partir de l'image (puisqu'il veut appliquer la taxonomie au sol à l'image). Il active ces procédures avec des paramètres choisis par lui, en fonction de sa connaissance du domaine, puis analyse les résultats, les accepte ou les rejete, en se basant sur des critères qui lui sont propres. Il construit ainsi des séquences (ou chaînes) de traitements. Si il en construit plusieurs, il choisira celle qui lui semblera la meilleure.

Cette capacité de production d'une chaîne de traitement est donc essentiellement humaine. Dans le cas de la cartographie à partir de données obtenues par télédétection, cette intervention humaine est incontournable car on prend en compte des zones d'ensoleillement différent, ainsi que des paramètres ad hoc (altitude, pente,...) pour affiner la classification.

Pour cet ensemble de raisons, il a été décidé la mise au point d'un système expert afin de répondre aux besoins de simplicité, de flexibilité et d'adéquation entre les données, le but poursuivi, et les actions menées.

B] CIME.1 : PREMIERE VERSION DU SYSTEME CIME

1 Présentation générale

La première version de CIME devait surtout permettre de juger de l'intérêt d'un système expert pour palier à l'impossibilité de résoudre le problème de façon entièrement procédurale. CIME.1 est bâti autour d'un moteur d'inférences fonctionnant en chaînage mixte. C'est un système de production dont les règles gèrent la recherche en agissant sur le contrôle, par le biais d'actions en conclusion ou en prémisse.

Dans les paragraphes suivants, les points importants de CIME.1 sont présentés de façon plus ou moins générale : pour plus de précisions, nous vous invitons à consulter le rapport de Monsieur Monjanel [Monjanel87], le concepteur de CIME.1

2 Principe de fonctionnement

Afin d'élaborer une carte de la région concernée par l'image, des traitements doivent être effectués pour obtenir une partition des pixels. La démarche supervisée qui est adoptée ici, consiste à identifier des éléments dans une image, dont on a une connaissance à priori, en guidant la recherche vers le type d'éléments recherchés. Si la méthode d'identification adoptée permet d'identifier correctement les éléments (on procède notamment par comparaison), elle est jugée bonne. Les éléments servant à éprouver les différentes méthodes sont appelés ici des **zones d'entraînement***.

Cette partition de l'image initiale se fait à l'aide de **segments***, produits par programme, et qui contiennent la base sur laquelle s'effectuera la classification des pixels (c'est un paramètre jugé discriminant : l'altitude, la radiométrie ...), et les contraintes permettant leur classification effective (on dit alors que l'on applique ces segments à l'image). Après avoir procédé à une première classification des pixels, on essaie alors d'affiner cette classification, en produisant d'autres segments prenant en compte une autre base de partition. L'image résultant de cette classification est dite étiquetée, chaque pixel ayant pour étiquette, la classe de végétation qui lui a été attribué par application des segments.

Cette succession de productions de segments, suivis de leurs applications à l'image, constitue une **chaîne**. Une production de segments suivie de leur application constitue une **étape**. Il apparait clairement qu'une **chaîne** est en fait une succession d'**étapes**. Le nombre d'étapes dans une chaîne est variable. En effet, les étapes se succèdent dans le but d'affiner une partition. Si une nouvelle étape donne une image moins bien *"classée"* que celle de l'étape précédante, la

chaîne est arrêtée, et une nouvelle chaîne est lancée. Une session de CIME est elle même une succession de chaînes.

Pour juger du succès d'une étape ou d'une chaîne, il existe des critères mis en place par l'expert, et formalisés sous forme de règles qui sont appliquées dès la fin d'une étape. Ces critères sont au nombre de deux : **"l'entropie"**, et le **nombre de pixels bien classés.** L'entropie est issue d'un calcul, et dénote le "désordre" résiduel après la classification. Le nombre de pixels bien classés s'obtient par comparaison de l'image classée avec les zones d'entraînement dont on connait à priori la classe de végétation. Un succès équivaut à une entropie minimale pour un nombre maximal de pixels bien classés. Ce sont ces deux critères qui sont utilisés pour départager les chaînes, et sélectionner la meilleure. Les étapes quant à elles, ne sont jugées qu'à leur entropie qui doit être inférieure à celle de l'étape précédante.

3 Les points importants

3.1 La mise en place des traitements numériques

Puisque ces traitements sont effectués suivant une seule méthode (la discrimination non paramétrique), ils ont été implémentés sous forme de procédures FORTRAN qui sont appelées depuis le système. Ces procédures sont connues par CIME.1 en tant que *procédures externes*, dont l'activation est explicitement demandée dans les règles de production.

3.2 La mise en place de la stratégie suivie

Ici, les règles de production ont un rôle particulièrement important, puisque ce sont elles qui guident la recherche. En effet, elles décident de l'enchaînement des étapes et des chaînes, provoquent l'appel des procédures externes, et effectuent les réinitialisations et les mises à jour par l'appel de procédures dites *internes* (écrites en PROLOG). Ce sont aussi les règles qui comportent les critères de satisfaction pour les chaînes et les étapes.

En conclusion, la stratégie adoptée dans CIME.1 est entièrement mise en place et contrôlée par les règles et les actions (internes et externes).

3.3 Les connaissances représentées

Ces connaissances sont de deux types : **iconique** et **conceptuel**. Le premier de ces types désigne essentiellement les pixels, tandis que le second désigne les unités de paysage que l'on étudie (ici, la végétation). Le but poursuivi est de mettre en correspondance ces deux types de connaissances afin d'aboutir à une carte dont les éléments iconiques sont étiquetés par des éléments conceptuels.

Dans CIME.1, ces deux sortes d'entités ne sont pas représentées au même niveau : les entités conceptuelles sont des **attributs** des entités iconiques (à ce titre, elles ne peuvent pas faire l'objet des mêmes descriptions). Il est important de signaler le rôle particulier de l'attribut **classe** qui représente dans CIME.1 les formes végétales possibles pour chaque pixel considéré. Cet attribut constitue un pivot entre les concepts utilisés et les pixels auxquels ils s'appliquent. C'est lui qui constitue le lien entre les deux *"mondes"* (celui de l'image et celui des concepts dans le domaine de la géographie) dont relève l'expertise.

Pratiquement, les éléments de travail sont :

* les **pixels** constituant les zones d'entraînement (puisque la démarche est supervisée) ;

* les "**segments**" ainsi nommés parcequ'ils permettent de partitioner les pixels, suivant qu'ils se plient ou non à certaines contraintes d'une part, et de leur attribuer ou non certaines classes de végétation d'autre part;

* la **méthode** (une seule ayant été envisagée au départ : dnp pour discrimination non paramétrique) devant produire les segments, à l'aide notamment de **paramètres** décrivant les pixels, et jugés discriminants. Ces paramètres proviennent de l'image (ex : la radiométrie), de l'étude préalable effectuée sur le terrain (ex : la pente), ou de calculs sur les données iconiques (ex : la texture).

3.4 L'implémentation

3.4.1 Le langage

Le langage choisi pour implémenter CIME.1 est le Quintus-Prolog. Ce langage a parfois facilité l'implémentation, et est à l'origine de certaines particularités du système.

Il s'agit tout d'abord de l'existence dans le système d'une représentation interne de l'image : les pixels constituant les zones d'entraînement. Lorsqu'une méthode permettant de cartographier l'image a été trouvée, la chaîne

de traitements qu'elle nécessite est appliquée à l'ensemble de l'image qui est externe au système. Cette application se fait par des procédures inconnues du système et activées par l'utilisateur. Ce choix d'une représentation partielle de l'image que l'on veut traiter, s'explique par le très grand nombre de pixels constituant une image (plusieurs milliers): les représenter tous poserait un problème d'optimisation des temps de traitements et de l'utilisation de l'espace mémoire. D'autre part, seuls des pixels d'entraînement sont nécessaires, puisqu'il suffit de trouver une méthode identifiant des formes de végétation pour des parties de l'image correspondant à des régions connues.

Une autre conséquence du choix de ce langage d'implémentation, est la nécessaire distinction entre procédures *internes* (écrites en QUITUS-PROLOG) et procédures *externes* (écrites en FORTRAN). La coexistence de deux langages de programmation provient du désir de réutiliser des procédures déjà écrites, en les intégrant au système.

Le dernier aspect de ce choix, est qu'il permet l'utilisation de nombreux prédicats, rendant très facile l'accès aux différentes informations utilisées par le système.

3.4.2 Structure des données

3.4.2.1 Les faits

Les faits sont représentés sous un prédicat : **VALIDE(FAIT)**. Chaque fait rend compte d'une relation du type **<Objet> <Attribut> <Valeur>**. Il existe deux sortes d'objets : les pixels qui sont représentés par des numéros, et le *contexte général* représentant l'image globale, qui est représenté par le signe '-'.

exemple: Certains faits dans CIME.1 sont :

```
valide(-,capteur,mss)
valide(-,ensoleillement_min,14)
valide(1,altitude,31)
valide(1,classe,[rhodo,sapin,chêne])
```

3.4.2.2 Les règles

Ce sont, comme nous l'avons signalé, des règles de production, qui ont pour forme :

Si
<*conditions*>
Alors
<*conclusions*>

où *conditions* et *conclusions* ont la même syntaxe, et sont d'une des formes suivantes :

* *attribut comparateur valeur*
* *attribut comparateur attribut*
* *action*

En conclusion, le seul comparateur autorisé est "=" qui signifie l'affectation. Les *Actions* sont "*internes*" (écrites en PROLOG et agissant sur la base de faits et le contrôle), ou "*externes*" (écrites en FORTRAN et produisant essentiellement les segments).

La distinction entre les pixels et le contexte, se fait au niveau des règles, qui sont dites **générales**, ou **locales**, suivant qu'elles s'adressent exclusivement ou non au contexte.

Certaines de ces règles sont paramétrées, afin d'éviter l'écriture de règles identiques, dans leurs conditions comme dans leurs conclusions, à une donnée près. Il n'y a qu'un seul paramètre représenté par la lettre "x", et il ne représente qu'un seul type de données : les variables utilisées pour produire les segments. A chaque instanciation de la variable, les règles paramétrées, et précédemment instanciées, sont détruites, puis instanciées de nouveau, par la nouvelle valeur de cette variable, et enfin utilisées pour l'inférence.

C] LES INSUFFISANCES ET LES DEFAUTS DE CIME.1

1 Une représentation limitée des connaissances

La représentation choisie dans CIME.1 ne permet de représenter qu'un seul niveau de connaissance qu'il s'agisse des entités iconiques ou des concepts associés. Un des inconvénients de ceci, est que l'on est obligé de toujours traiter l'ensemble des pixels constituant les zones d'entraînement, sans pouvoir focaliser l'étude, ou même, traiter uniquement les pixels non étiquetés, en laissant de côté ceux pour lesquels la forme de végétation a été reconnue.

D'autre part, la structuration des pixels en éléments iconiques plus généraux (régions connexes, ou d'objets plus généraux) n'est pas possible. Du point de vue conceptuel, ces

limitations se font aussi ressentir : il n'est pas possible à l'expert de décrire les relations entre les différents concepts qu'il utilise, car ce ne sont que des attributs, et ils ne sont pas descriptibles.

2 Un objectif limité

Les options choisies dans CIME.1, notamment la représentation des connaissances, ne permet pas à l'utilisateur d'aller plus loin que la production d'une chaîne de traitements permettant de cartographier l'image. L'application même de ces traitements ne peut se faire dans le système, alors qu'il serait intéressant de pouvoir travailler sur la carte produite dans le but de focaliser l'étude, ou de la compléter. En effet, dans le cas où l'on veut affiner la cartographie d'un thème, on ne raisonne plus à partir de "toute l'image", mais à partir d'un sous ensemble déjà partiellement interprété (c'est une focalisation thématique).

3 Une stratégie de recherche limitée

La gestion des étapes et des chaînes par les règles, fait qu'en cas d'échec, aucun mécanisme de reprise n'est possible : la chaîne est abandonnée. Il aurait été souhaitable de pouvoir remettre en cause la dernière étape de la chaîne ayant conduit à l'échec. Ce rôle que jouent les règles, impose un déroulement très procédural de la recherche, et fait qu'il n'y a pas toujours une parfaite adéquation entre les données et la stratégie de recherche.

La limitation de la stratégie de recherche s'exprime aussi dans le fait qu'une seule *méthode* est envisagée pour produire les segments, et toujours suivant une démarche *supervisée*.

4 Une implémentation inefficace

Plusieurs points sont à relever dans l'implémentation de CIME.1. Il faut cependant préciser que lors de la mise au point de cette première version du système, l'optimisation n'était pas prioritaire, et la complexité de l'ensemble du projet, faisait que le but essentiel était d'obtenir un système qui puisse, dans une démarche reproductible, produire un ensemble de traitements permettant de cartographier un thème à partir d'une image satellitaire.

4.1 Les mises à jour dans la base de connaissances

A chaque nouvelle étape, les données utilisées dans le système sont réinitialisées, car leur grand nombre interdit toute duplication. Par ailleurs, au cours de l'étape,

l'application des segments nécessite, pour tous les pixels, une nouvelle instenciation de la valeur de l'attribut *"classe"*. Le processus classique de mise à jour consiste à enlever de la base de connaissances les faits correspondant aux anciennes valeurs, et à insérer ceux correspondant aux nouvelles, afin de toujours conserver une base de connaissances cohérente.

Il est important de remarquer que le principe de fonctionnement impose la nécessité de toujours avoir une base de connaissances cohérente. Cela signifie qu'aucune contradiction ne doit exister dans les faits contenus dans la base (par exemple l'existence simultanée des faits X=1 et X=2, constitue une contradiction). La conséquence d'une contradiction dans un système déductif tel que CIME, c'est que des inférences pourraient avoir lieu sur des données erronées (ie. participant à la contradiction), ce qui entraînerait la production d'autres faits, eux même erronés.

Dans CIME.1, la procédure chargée de ces mise à jour, **MISEAJOUR**, procède au retrait de tous les faits de la base de connaissance, et en se basant sur les dates de validation, par la suite, valide de nouveau les faits ne devant pas être remis en cause. Etant donné le nombre important de faits devant à chaque fois être "revalidés" et la fréquence de ces mises à jour, le temps d'exécution de cette procédure est particulièrement important.

4.2 La gestion des règles paramétrées

Comme il été souligné précédemment, les règles génériques sont instanciées dès que la variable est affectée d'une valeur (c'est la variable jugée discriminante pour la production des segments). A chaque nouvelle instanciation de la variable (ie. au début de chaque étape), les règles précédemment instanciées sont impérativement détruites, et remplacées par les règles nouvellement instanciées. La gestion qui s'en suit est complexe et relève de la compilation de règles.

En effet, il est nécessaire pour l'inférence, que des informations précises sur les attributs en conditions et en conclusions de règles, soient disponibles. Aussi, chaque nouvelle instanciation de règle implique-t-elle la mise à jour de ces informations. Or, d'une chaîne à l'autre, les mêmes variables servent à instancier les règles paramétrées : la destruction des règles à chaque nouvelle instanciation, fait perdre le bénéfice de toutes les mises à jour précédemment effectuées.

4.3 Les actions internes, le contrôle, et la stratégie

Il est à noter une très grande imbrication entre les actions internes, le contrôle et la stratégie de recherche. En effet, les actions dites internes et chargées de la gestion de la base de faits (en particulier,la procédure de mise à jour), agissent sur le contrôle en provoquant des inférences en marge de celles nécessitées par la recherche, et en modifiant les informations gérées par l'inférence, pour en diriger le comportement. On note aussi que le contrôle lui même est adapté à la stratégie adoptée, dans la mesure où des traitements spécifiques à la recherche sont en partie assurés par le moteur d'inférences, qui, pour ce faire, se base sur la nature des éléments qu'il traite.

Il n'est pas intéressant que de telles interdépendances existent car elles sont un obstacle à la généralité du système, et par conséquent, elles n'autorisent pas son utilisation dans d'autres conditions. En ce sens, CIME.1 est un système figé.

D] CIME.2 : LA NOUVELLE VERSION DE CIME

1 Pourquoi une nouvelle version de CIME ?

Il a été souligné précédemment que CIME.1, la première version de CIME, devait faire les preuves de faisabilité de l'implémentation, sous forme de système expert, d'une démarche en télédétection, ayant pour but initial la cartographie.

Ce but a été atteint, et l'expérience a permis de révéler les difficultés inhérentes à une telle implémentation. Ces difficultés se traduisent par un aspect encore très procédural, imposé au système par les règles, une imbrication trop forte entre le contrôle et la thématique. Le désir et la nécessité d'aboutir à un programme qui marche, ont contraint les concepteurs à adopter une optique restreinte, ne permettant que la seule production d'une chaîne de traitements destinée à effectuer une classification plus ou moins satisfaisante d'une image.

L'expérience acquise lors de l'élaboration de CIME.1, a permis alors d'envisager la mise au point d'un nouveau système plus large, plus général, et plus performant.

2 Les projets de CIME.2

CIME.2 se doit de résoudre les problèmes laissés en suspend dans CIME.1, et s'étendre afin de permettre un travail plus large et plus complet dans le domaine de la cartographie thématique. La première version de CIME effectuait des travaux

(menant à la classification de l'image), qui, dans CIME.2, ne constitueront plus qu'une phase.

2.1 Une représentation plus complète des connaissances

Il sera possible dans CIME.2 de représenter plus complètement les connaissances de l'expert. Les connaissances à représenter sont **factuelles** (les données de base telles que l'image, ou tout autre fait dûment établi, relevant en général du domaine de l'expertise), ou **procédurales** (il s'agit surtout des règles de production, et des traitements sur les données, liés au domaine d'application), et concerneront des entités de nature **iconique** ou **conceptuelle**.

Par ailleurs, il n'est pas exclu que des connaissances **prototypiques** (c'est à dire définies directement par la valeur de leurs attributs) soient utilisées dans un but stratégique. En effet ces données prototypiques permettent la reconnaissance directe des entités qu'elles décrivent. Dans CIME, elles permettent aussi d'établir le lien entre les entités iconiques et conceptuelles.

Ainsi, le niveau de description et d'utilisation des connaissances sera-t-il variable, et la base de connaissances aura, exprimant sa richesse et sa diversité, un caractère hétérogène, puisqu'elle contiendra des éléments de nature différentes; dans ce cadre, les éléments sont appelés des **objets***.

2.1.1 Les entités iconiques

Il est important, pour les études envisagées, de pouvoir disposer de moyens permettant d'extraire de la carte produite, des éléments significatifs. En effet, dans le domaine de la géographie, le pixel, élément de base d'une image, n'est pas représentatif des réalités observées sur le terrain. De plus, il est plus proche du sens commun, dans ce domaine, de raisonner par regroupements, comparaisons et extraction d'ensembles, dont les éléments sont caractérisés par des propriétés communes.

Sous l'une ou l'autre des deux formes de représentation, on pourra décrire des éléments de l'image qui soient plus proche d'une représentation iconique des concepts étudiés, et ayant une signification relative au domaine (ici la géographie). Ainsi pourra t'on effectuer des regroupements de pixels afin d'obtenir des régions ou des objets plus généraux (par exemple : on pourra créer des régions classées forêt, en effectuant un regroupement des pixels ayant pour végétation la forêt). Ces éléments pourront être agrégés à leur tour pour donner d'autres éléments de type "objets généraux". Il sera alors possible de mettre en place une **hiérarchie structurelle** des éléments de l'image. Cette hiérarchie sera construite de façon ascendante, suivant les regroupements effectués.

Cet élargissement dans la représentation des connaissances, permettra de mettre en évidence et d'utiliser au mieux, les relations de type **topographique** et **structurelle**, liant les différents constituants de la carte.

2.1.2 Les entités conceptuelles

Au niveau conceptuel aussi, la représentation des connaissances s'enrichira : les concepts seront, comme dans CIME.1 des attributs d'entités iconiques, mais dans CIME.2 ils deviennent descriptibles.

Tout d'abord, allant de pair avec les nouveaux éléments construits, de nouveaux concepts seront intégrés au système. Tout comme les entités iconiques, les entités conceptuelles pourront être décrites au moyen d'une hiérarchie, il s'agira alors d'une **hiérarchie conceptuelle**, dont la structure sera basée sur les relations de généralisation/spécialisation liant les différents concepts entre eux.

Etant donnée la nature différente de ces relations, comparativement aux entités iconiques, cette hiérarchie sera construite de façon descendante. Contrairement à la hiérarchie structurelle, qui ne peut être construite qu'au cours de l'étude conduisant à l'élaboration de la carte (suivant le but poursuivi, les éléments dégagés ne seront pas toujours les mêmes), la hiérarchie conceptuelle pourra être mise en place par l'expert comme une donnée initiale, sous forme de faits.

2.2 Des outils performants

2.2.1 Des mots clé

Afin d'accéder aux informations, il sera mis à la disposition de l'expert et de l'utilisateur des **mots clé**. Il permettront essentiellement de désigner la nature des objets utilisés, et de référencer des éléments dans les hiérarchies construites.

2.2.2 Des primitives

Par ailleurs, dans le but de permettre et faciliter l'étude plus poussée de l'image et de la carte produite, il existera un interface avec *"l'extérieur"* plus sophistiqué. En effet, cet interface devra permettre, en plus de l'appel aux procédures externes, de propager dans une image (il s'agira de la représentation interne, ou de l'image -ou de la carte- qui demeurent extérieures au système) les modifications intervenues dans l'autre. Ces répercutions se feront au moyen de **primitives** qui feront partie de l'interface.

2.2.3 Des fonctions propres au système

D'autre part, puisque de tous les résultats des traitements en télédétection, la carte est la seule à rendre compte de l'espace, il sera utile de pouvoir utiliser les relations spatiales (proximité, superposition, position ...) liant les éléments iconiques entre eux. Pour cela des procédures appelées **fonctions propres du systèmes** seront mises en place. Elles se comporteront comme des fonctions booléennes, ce qui permettra leur utilisation en tant que comparateur dans les conditions de règles.

2.2.4 Un paramétrage plus général des règles

Enfin, la nature très différente des éléments de la base de connaissances, conduit à la mise en place de la possibilité d'un paramétrage plus général des règles de production. La distinction entre *règles locales* et *règles globales* ne sera plus nécessaire, puisque les outils présentés (notamment les mots clé), permettront la désignation et la description précise des éléments concernés par les conditions et les conclusions de ces règles. Ces règles seront appliquées par le moteur d'inférences, tant qu'il sera possible de trouver dans la base de connaissances des éléments vérifiant les conditions (par exemple, si une règle concerne un objet dont le type est précisé, avec des conditions portant sur certains de ses attributs, elle sera déclenchée tant que des éléments du type requis vérifieront la propriété sur les attributs mentionnés).

Ceci autorisera l'expert à s'adresser à des éléments très différents dans une même règle (par exemple, dans une même règle, en définissant en conclusion une nouvelle entité iconique, on pourra si besoin est, créer un nouveau concept s'y rapportant, et rattacher celui-ci à la hiérarchie conceptuelle). De plus, cela rendra plus explicites et plus compréhensibles les règles ainsi écrites.

2.3 Un objectif plus large

Le schéma plus général de travail, devra permettre de classer l'image, d'en dégager des éléments particuliers sur lesquels focaliser l'étude, et d'étudier de façon plus précise ces éléments et les phénomènes qui les concernent.

exemple : Pour les régions montagneuses du Nepal, qui sont les régions étudiées par CIME, il peut être intéressant ou nécessaire d'étudier les phénomènes de déforestation. Pour ce faire, on classera l'image suivant le principe décrit précédemment, on déterminera les zones forestières et leur bordures, puis on focalisera l'étude sur ces deux types de zones.

En plus de la diversité des traitements, des méthodes et des démarches pouvant être suivies, CIME.2 pourra être utilisé pour des régions autres que les zones montagneuses du Nepal. Demeurant un système de traitement d'images satellitaires, CIME.2 pourra être très diversement utilisé, pour peu que les données (l'image), l'expertise (la connaissance du problème, des stratégies et des possibilités de résolution), et les traitements (les procédures informatiques adéquates pour l'utilisation des données) soient disponibles.

La généralité de CIME.2 se traduira aussi, et surtout, par une plus grande indépendance entre l'expertise et le système d'inférences, autrement dit, entre la connaissance et le contrôle. Pour cela, les algorithmes qui seront implémentés seront généraux, entièrement basés sur la syntaxe, et non sur la signification, des données qu'ils auront à traiter.

2.4 Une implémentation mieux adaptée

Nous avons vu que l'imbrication entre le contrôle et la thématique était imputable aux règles qui, par les actions qu'elles déclenchaient, et la gestion des étapes et des chaînes qu'elles assuraient, contribuaient à compliquer le comportement du système, et à en limiter l'extension.

Il a donc été décidé que CIME.2 comporterait une implémentation adaptée à la gestion des mises à jour et du contrôle de la stratégie. Cette implémentation se traduira par la mise en place d'algorithmes spécifiques aux problèmes soulevés, et suffisamment généraux pour permettre l'adaptation du système aux différentes études qu'il sera amené à effectuer.

3 Les problèmes à résoudre

Dans ce dernier paragraphe, nous faisons un bref rappel des aspects importants de CIME.2, et nous ne faisons que souligner les problèmes rencontrés dans son élaboration. Ces problèmes sont liés au domaine d'application et aux objectifs fixés pour le nouveau système : les solutions proposées dans les chapitres suivants, tiennent compte de ses spécificités.

3.1 Les hypothèses

Nous avons vu précédemment qu'une session consistait à sélectionner une chaîne de traitements parmi plusieurs, et que ces chaînes étaient elles même une succession d'étapes. La meilleure des chaînes est celle pour laquelle le choix des paramètres discriminants est pertinent. Puisque plusieurs combinaisons sont possibles (l'ordre d'introduction de ces

paramètres étant important), les choix à effectuer doivent être supervisés. De plus, dans le cas où plusieurs méthodes sont envisageables pour produire les éléments nécessaires à la classification, le nombre de possibilités peut devenir très grand (en fait, ce nombre dépend directement du nombre de paramètres pris en compte par les méthodes).

Lorsque l'on met en place une chaîne, on **suppose** donc que les choix effectués sont bons, mais s'ils ne l'étaient pas, il faudrait pouvoir revenir sur ces choix, et en effectuer d'autres. Il a donc été décidé de gérer les chaînes et les étapes pour leur mise en place et leur remise en cause, en tant qu'**hypothèses**. Une hypothèse dans CIME.2 concernera donc la *méthode* de classification, et les *arguments* de cette méthode. Le premier problème à résoudre est donc la gestion des hypothèses : leur émission, l'évaluation de leur pertinence, et leur remise en cause.

3.2 La non monotonie

Les traitements effectués dans CIME, ont pour résultats, l'étiquetage de l'image et de ses composants (c'est à dire l'attribution de valeurs aux attributs des éléments). Quel que soit le but, ces traitements successifs se font sur les mêmes données : il n'y a pas duplication des données de base. On est donc amené à stocker les résultats sous forme concise et explicite (par exemple ,d'une chaîne à l'autre on enregistrera les méthodes utilisées pour arriver à l'image étiquetée, plutôt que l'ensemble de l'image étiquetée). Ceci a l'avantage de réduire la masse de données utilisées, mais cela implique surtout de nombreuses réinitialisations, pour permettre aux traitements successifs de se dérouler correctement. La réinitialisation des données est donc capitale dans CIME.

D'autre part, CIME.2 utilisera un raisonnement hypothétique, ce qui signifie que les choix effectués peuvent ne pas se révéler pertinents. A un certain point de l'inférence, il faut donc pouvoir arrêter le mécanisme de recherche, revenir en arrière, et "*effacer*" ce qui a été fait.

Il est évident que le problème posé par la non monotonie n'est pas le retrait en lui même, mais les nécessaires mises à jour qui l'accompagnent. Le second problème à résoudre est donc la gestion de ces retraits et la mise en place d'un mécanisme de mise à jour. Nous avons déjà signalé que la gestion de ces mises à jour sera assurée par un algorithme spécialement dédié à ce problème.

3.3 La représentation des connaissances

Généralement, la représentation des connaissances est un problème difficile: il faut savoir ce qu'il est nécessaire de représenter, et surtout, la façon de le représenter. Dans CIME, l'expertise se situe à deux niveaux : le thème, et la télédétection. Donc, il s'agira surtout de l'expression iconique du thème, ce qui nécessite des informations issues de l'image d'une part, et les connaissances du thématicien d'autre part.

Au niveau de l'expert, ces données ne relèvent pas seulement de la description, et sont souvent conceptuelles. La difficulté est alors de pouvoir représenter ces connaissances afin que les informations qu'elles contiennent soient facilement accessibles et utilisables.

Nous avons signalé les possibilités de représentation pour les connaissances, qui seront offertes dans CIME.2 : l'agrégation d'éléments iconiques, la mise en place d'une hiérarchie structurelle, les possibilités de référencer les éléments de cette hiérarchie, la caractérisation des éléments manipulés, la mise en place des relations entre les concepts, la description de ces concepts (considérés à leur tour comme des objets)... Dans le souci de respecter le caractère général du système, il est évident que tout ceci devra être formulé au moyen d'une syntaxe qui devra être simple, facile à utiliser et suffisamment générale pour éviter la multiplicité des "cas particuliers", aussi bien pour ce qui est de la représentation, que de l'utilisation.

CHAPITRE II

LES PRINCIPES ALGORITHMIQUES

A] GESTION DES HYPOTHESES : LE CHOIX DE C.H.(Choix-Hypothèses)

C.H., est une adaptation d'un algorithme plus général : A.D.D.B. expliqué au chapitre 2. Mais que signifie ADDB ? Pour l'instant nous dirons seulement ce que signifient ces initiales : Assumption-based Dependency-Directed Backtracking, le principe qui se cache derrière sera exposé en détail par la suite.

Rappelons que CIME.1 est entièrement piloté par les règles, qu'une seule méthode y est envisagée, et que les chaînes et les étapes sont entièrement mises en place par les règles. Il n'y a donc pas véritablement de choix puisque l'invocation des chaînes et des étapes est prédéterminée. Dans CIME.2, les étapes et les chaînes seront gérées comme des hypothèses.

1 Les autres méthodes

Le problème de choix et de retour sur un choix relève de la gestion d'hypothèses : émission d'une hypothèse, vérification de sa validité et de sa pertinence, remise en cause de cette hypothèse en cas d'échec.

Ce problème a été résolu dans d'autres systèmes par des moyens différents. Nous parlerons de SNARK [Laurière88], de TMS [Doyle79] et [Doyle83], de ATMS [deKleer86a], et bien sûr de ADDB [deKleer86b] qui traitent de gestion d'hypothèses; mais aussi de MYCIN [Shortliffe76], et du principe des heuristiques qui offrent d'autres approches du problème des choix. Il reste à ajouter que bien souvent, le raisonnement hypothétique se fait lorsqu'il y a des informations incomplètes (c'est le cas de la plupart des systèmes cités).

1.1 MYCIN

MYCIN est un système destiné à proposer un diagnostic, ou des traitements, dans le domaine des maladies bactériennes du sang. Afin de pallier les nuances qu'engendrent forcement le diagnostic, et les différentes démarches qui y conduisent, les hypothèses émises se traduisent par un coefficient de vraisemblance, qui est le reflet de la validité que l'expert accorde à ses diagnostics intermédiaires (assimilables à des hypothèses), et même terminaux. Ainsi, dans MYCIN, à chaque fait est associé un coefficient de vraisemblance compris entre -1 et 1, chaque règle tient compte de ces coefficients, et en produit d'autres qui seront attribués à ses conclusions.

exemple : Une des règles de MYCIN est :

SI 1) Le site de la culture est le sang, et
1) La coloration GRAM de l'organisme est GRAM-, et
2) La morphologie de l'organisme est de type bâtonnet, et
3) L'aérobicité de l'organisme est anaérobique, et
4) le seuil d'entrée de l'organisme est GI

ALORS Il existe *une évidence fortement suggestive (0,9)* que l'identité de l'organisme soit bactéroide

Ici, l'expert considère la conclusion comme une *"évidence fortement suggestive"*, ce qu'il traduit par le coefficient 0,9 (90%).

Ceci donne nécessairement lieu à des calculs et combinaisons de coefficients qui sont compliqués. On ne peut vraiment parler de gestion d'hypothèses dans le cas de MYCIN, puisque ne sont explorées que les branches de l'arbre de recherche donnant aux faits produits les plus gros coefficients. Les conclusions de la recherche sont elles aussi données avec un coefficient de vraisemblance qui traduit l'appréciation de l'expert vis à vis du ou des diagnostic (s) proposés.

1.2 LES HEURISTIQUES

Une heuristique est associée à une fonction d'évaluation qui, à un noeud de l'arbre de recherche, en évaluant les différentes possibilités, attribue des coefficients aux différentes branches de l'arbre. La stratégie heuristique consiste alors à ordonner les choix suivant les résultats de la fonction d'évaluation. Cette évaluation se fait à une plus ou moins grande profondeur, et à ce titre ne permet qu'une estimation limitée des conséquences des choix effectués, et donc de possibles erreurs dues à un manque de recul. Lorsque la fonction d'évaluation est une fonction de gradian, cette stratégie n'est pas **complète***.

D'autre part, l'évaluation nécessite le développement de l'arbre de recherche en largeur, à une profondeur déterminée. Lorsque les données sont nombreuses et les possibilités non moins restreintes, appliquer une fonction d'évaluation à chaque noeud de l'arbre devient très coûteux en temps et en traitements.

Ces dernières remarques expliquent les raisons pour lesquelles le principe des heuristiques n'a pas été retenu pour gérer les hypothèses dans CIME.

1.3 SNARK

SNARK peut être assimilé à un langage de programmation puisqu'il met à la disposition de l'utilisateur de nombreuses commandes utilisables dans des règles de production, le raisonnement sous-jacent étant le chaînage avant [Laurière86]. Cependant, il n'y a pas à proprement parler de mécanisme de gestion d'hypothèses dans SNARK. En effet l'émission d'une hypothèse se simule par ajout d'une assertion, le retour sur une hypothèse se fait par le biais d'une commande : REVENIRSUR(assertion).

C'est donc à l'utilisateur de gérer ses hypothèses, par le biais de règles de productions, pour ce qui est de leur émission comme de leur remise en cause. De plus dans SNARK la détection des incohérences n'est pas assurée par le système : c'est aussi à l'utilisateur de s'en charger.

Dans CIME.1, c'est ce principe d'une gestion par l'utilisateur, au moyen de procédures, que l'on a tenté d'implémenter. Les raisons de l'abandon de cette implémentation dans CIME.2 sont exposées au chapitre précédant.

1.4 TMS

Le Truth-Maintenance System proposé par Doyle permet, grâce à une structure particulière des faits, d'émettre et de gérer des hypothèses. Cette gestion est indépendante de l'inférence puisque TMS est séparé du module déductif. Pour cela, il utilise les notions de **justification correcte** et de **statut** pour chaque fait. Ces notions sont liées : un fait correctement justifié aura un statut **IN**, sinon il aura un statut **OUT** (le statut **OUT** ne traduit pas le *NON* sémantique). Les hypothèses sont alors issues de règles utilisant une logique par défaut :

exemple : **SI** OUT P **ALORS** Q

qui signifie :

"en l'absence d'informations sur P supposer Q "

Dans TMS, chaque fait F possède une liste de justifications. Une justification de F est la liste des faits IN (IN-justification) ou OUT (OUT-justification) dont dépend F. Une hypothèse est un fait dont chaque justification contient au moins une assertion de statut OUT (OUT-justification). Toutes ces informations sont rattachées à chaque fait F dans une structure de données mettant en évidence, par le biais des justifications, les liens de dépendance entre les données (data dependencies), et qui est la suivante :

(F, LISTE DES JUSTIFICATIONS, LISTE DES JUSTIFIES, LISTE DES SUPPORTEURS, LISTE DES SUPPORTES, STATUT)

où :

-LISTE DES JUSTIFICATIONS : liste des faits ayant permis par leur présence (IN-justifications), ou par leur absence (OUT-justification), de dériver F.

-LISTE DES JUSTIFIES : Liste des faits ayant F dans leurs justifications.

-LISTE DES SUPPORTEURS : Liste des faits issus de la liste des justifications, et choisis comme support principal.

-LISTE DES SUPPORTES : Liste des faits ayant F dans leur liste des supporteurs.

-STATUT : A pour valeur **IN** ou **OUT**, suivant que le fait est ou non correctement justifié.

Avec cette structure, les faits initiaux (ou prémisses) ont une liste des justifications vide. Pour qu'un fait F ait un statut à IN, il faut qu'il ait au moins une chaîne de justifications non circulaire, partant de F, et aboutissant à un fait initial ou à une hypothèse. D'où la notion de WFS, pour Well Founded Support, dont le supporteur est un représentant. Pour un fait de statut IN, plusieurs chaînes de justifications peuvent exister. Lorsque le statut d'un fait est à OUT, la liste des supporteurs contient un représentant de chaque élément de la liste des justifications.

TMS assure aussi le maintien de la cohérence de la base de faits. Les contradictions sont détectées par le module déductif, qui en informe TMS en lui fournissant la liste des faits (justifications) engendrant la contradiction. Parmi eux, TMS en choisit un qui soit une hypothèse, change le statut de cette hypothèse, et rétablit la cohérence de la base de fait suivant ce nouveau statut. La liste des faits ayant conduit à la contradiction est enregistrée dans un "no-good set" afin d'être évitée par la suite.

TMS permet donc l'émission d'hypothèses et en assure la gestion par une stratégie de type *backtrack* (ou retour arrière). Il permet de trouver une solution, si il en existe, dont on soit sûr de la consistance vis à vis des hypothèses faites.

Une telle gestion est intéressante, puisqu'elle gère à la fois les hypothèses (émission et remise en cause), et la cohérence de la base de faits, évitant ainsi à l'expert l'écriture de règles de gestion, toujours délicates à mettre en oeuvre.

L'inconvénient de ce principe est que les contradictions peuvent être détectées pour un ensemble non minimal de faits. Des inférences inutiles sont donc effectuées, dans un environnement qui se révèlera contradictoire, mais qui aurait pu être découvert comme tel plus tôt, et donc évité. Cette "contre-performance" peut être néfaste dans le cas où, comme dans CIME, le nombre des données manipulées est grand et les traitements coûteux. D'autre part, la remise en cause d'une hypothèse, lorsqu'une contradiction est détectée, se fait sans directives : c'est la première hypothèse rencontrée dans l'environnement contradictoire qui est remise en cause. Ce manque de contrôle dans le retour arrière, n'est pas souhaitable dans CIME où la nature même des hypothèses (méthode et arguments), et de leurs utilisations, nécessite que les retours sur les choix soient contrôlés.

1.5 ATMS

ATMS (Assumption based Truth-Maintenance System) proposé par De Kleer [deKleer86a], sur un principe similaire à celui de TMS, offre une gestion des hypothèses où il n'y a pas véritablement de choix ni de retour sur ces choix. En effet toutes les hypothèses sont envisagées en **parallèle** avec une stratégie de **recherche en largeur**. On ne retrouve pas la notion de statut utilisée dans TMS, mais celle de justification associée à celles **d'environnement** et de **label**. Ici, les hypothèses sont supposées connues dès le départ, et repérées comme telles : ce sont les **hypothèses de base**. Un environnement est un ensemble d'hypothèses de base dont le fait dépend directement. Les justifications sont, comme dans TMS, l'ensemble des assertions dont dépend le fait. Un label est un ensemble d'environnements, c'est lui qui remplace la notion de statut de TMS.

Un fait F dans ATMS a donc la structure suivante :

(F, LISTE DES JUSTIFICATIONS, LABEL)

exemple : **SI** A **ET** B **ET** C **ALORS** F

avec A et C hypothèses de base, un environnement de F est donc la liste (A,C), et sa justification la liste (A,B,C). Si, en plus, on a :

SI H **ALORS** F

avec H hypothèse de base, un nouvel environnement de F est (H) et son autre justification est aussi (H).

Le fait F s'écrit donc :

(F , ((A,B,C) , (H)) , ((A,C) , (H)))
fait justifications label

F est vrai pourvu que l'on soit dans l'environnement (A et C) ou dans l'environnement (H), et que toutes les assertions d'une justification de F contenant l'un de ces environnements soient vérifiées.

ATMS est un système monotone : il n'y a pas de retour arrière. Les nouveaux faits sont transmis à ATMS par le module déductif avec leurs justifications. Le label du nouveau fait est calculé en faisant la conjonction des labels des faits présents dans sa justification. Puisque chaque label est une disjonction d'environnements, le calcul d'un nouveau label nécessite une implémentation qui permette d'optimiser ces opérations ensemblistes, et d'éliminer les redondances possibles dans les résultats. Dans ATMS, les hypothèses sont représentées par un tableau de bits (une hypothèse est associée à chaque bit) et les opérations ensemblistes deviennent des opérations booléennes sur les tableaux de bits (AND pour l'intersection, OR pour l'union, AND et le complémentaire pour l'inclusion).

Ici aussi, le contrôle de la cohérence est effectué et optimisé en :

- commençant par effectuer les inférences dans les environnements les plus restreints, et en retardant au maximum l'introduction d'une nouvelle hypothèse.

- mémorisant les environnements contradictoires dans des "no-good". Puisque ces environnements contradictoires sont minimaux, tout sur-ensemble de ces environnements est détecté comme contradictoire.

Ainsi, contrairement à TMS, dans ATMS les inférences sur des environnements devant se révéler contradictoires sont évitées.

Remarque : On peut noter que les deux faits suivant : (X=0,J1,L1) et (X=2,J2,L2), peuvent coexister dans la base de faits sans qu'il y ait contradiction.

La stratégie de recherche en largeur, adoptée dans ATMS, assure de trouver toutes les solutions, si il en existe, tout en garantissant une bonne gestion des contradictions. Cependant, il peut se révéler gênant que toutes les solutions soient recherchées systématiquement, lorsque qu'une solution suffit. Mais l'inconvénient majeur de la recherche en largeur, c'est que le nombre de possibilités doit être restreint, ce qui n'est pas le cas de CIME.2 (cf. chapitre I, D] § 3.1).

2 ADDB

2.1 Présentation générale

Ce principe proposé par De Kleer et Williams [deKleer86b], consiste en un mélange de TMS et de ATMS dans le but de conserver les avantages de chacun, tout en évitant leurs inconvénients respectifs : il s'agit donc de contrôler le retour arrière sur un choix.

ADDB, pour Assumption-based Dependency-Directed Backtracking, est en fait un algorithme permettant de procéder à un ensemble de choix qui soit **complet**, permettant ainsi l'obtention d'une solution valide. Pour cela, ADDB gère une pile dont les éléments sont appelés **disjonctions de contrôle**, et qui représentent des ensembles d'assertions incompatibles deux à deux. Chacune de ces assertions constituera une **hypothèse du contexte.** L'utilisation de ADDB suppose que toute solution ne peut être valide que si elle se réalise dans un environnement où exactement **une** assertion, issue de chaque disjonction de contrôle, a été prise comme hypothèse (c'est ce qui a été appelé ensemble complet de choix).

Ces disjonctions ont pour forme : CONTROL(*C1,C2,C3,...*), où chaque C_i représente une assertion; C_i et C_j étant incompatibles pour i=/=j.

Les différentes hypothèses peuvent dépendre l'une de l'autre, aussi existe-t-il la possibilité de mettre en place, ou d'inhiber des disjonctions de contrôle, au cours de l'algorithme. Ceci se fait avec des règles de production identiques à celles utilisées pour les inférences.

exemple : Soit la règle C_i ==> ACTIVE(CONTROL(*A,B,C*)); PASSIVE(CONTROL(*X,Y*))

qui signifie :

SI l'assertion C_i est validée,

ALORS la disjonction de contrôle CONTROL(*A,B,C*) devient active, et est à ajouter à la pile;

ET la disjonction de contrôle CONTROL(*X,Y*) devient passive, et est à enlever de la pile.

2.2 Les algorithmes

Le principe de l' algorithme général de ADDB est le suivant :

a) Partir d'un ensemble d'hypothèses initialement vide,
b) Si la pile est vide, l'environnement est complet =>le tester et éventuellement inférer
c) Dépiler la disjonction en sommet de pile
d) Si la disjonction n'a plus d'élément =>retourner au dernier choix effectué
e) Prendre le premier élément de cette disjonction et l'ajouter à l'environnement courant
f) Si ce nouvel environnement est consistant, recommencer en b)
g) Sinon retourner au dernier choix effectué.

Ce premier algorithme ne fait pas apparaître la gestion de la pile, lors d'ajout ou de retrait de disjonctions; il expose seulement le principe de choix des hypothèses et du retour contrôlé sur ces choix. Cette gestion particulière des disjonctions de contrôle se fait comme suit :

h) Une disjonction passive devient active

Elle est ajoutée à la pile, une de ses assertions choisie de façon à ce que, ajoutée à l'environnement courant, le nouvel environnement obtenu soit consistant. Si une telle assertion ne peut être trouvée, l'environnement courant est marqué comme étant inconsistant, et la recherche continue tant que la pile ne change pas par ajout ou retrait de disjonction.

i) Une disjonction active devient passive

On dépile jusqu'à la disjonction de contrôle concernée. A chaque dépilement d'une disjonction, l'assertion correspondante est enlevée de l'environnement courant. Puis les disjonctions de contrôle demeurant actives sont remises sur la pile, tout en fournissant leur première assertion qui soit consistante avec le contexte.

j) Lorsque le contexte devient inconsistant, sans que la pile ait changé, le retour arrière s'effectue comme dans l'algorithme vu précédemment.

2.3 Gestion des "no-good"

Lorsque l'algorithme ADDB produit un environnement (environnement s'entend ici au sens où il est utilisé dans ATMS : comme un ensemble d'hypothèses de base), ce dernier est testé par un algorithme de type ATMS, qui vérifiera en même temps, suivant un ordre précis, la validité de l'environnement en lui appliquant les règles de production. Ceci est fait dans l'ordre chronologique de mise en place, en commençant par les

environnements les plus petits, dans lesquels des règles peuvent être appliquées. Si un environnement se révèle contradictoire, il est alors répertorié comme tel sous l'appellation de **"no-good"**.

exemple : Soit l'environnement terminal (A,§,1) (étant entendu qu'ADDB a procédé au choix de A, puis de §, et enfin de 1). La vérification se fera en appliquant les règles sur les environnements successifs suivants : (A) , (§) , (A,§), (1) , (A,1) , (§,1) , (A,§,1).

Afin d'optimiser la recherche des "no-good", des règles dites **d'hyper-résolution** sont mises en place. Ces règles permettent d'obtenir, par inférences à partir des "no-good" connus et des disjonctions de contrôle, tous les "no-good" minimaux déductibles.

Ces règles d'hyper-résolution s'écrivent comme suit :

(*Si* CONTROL*(A1,A2,A3,..Ap.)*
(*et pour tout i appartenant à E = {1,..,p},*
(*N_i est un nogood ,tel que A_i appartienne a N_i*
(*et pour tout j de E - {i},*
(*A_j n'appartienne pas a Ni*
(*Alors $U_k(N_k - (A_k))$ pour k appartenant à E, est un no-good*

Cette règle s'interprète comme suit :

Si *il existe une disjonction de contrôle, telle que : chacune de ses assertions soit un élément d'un "no-good"* **(chacun de ces "no-good" ne comportant qu'une seule assertion provenant de cette disjonction),**

Alors *l'union de ces "no-good", privée des assertions citées, constitue aussi un "no-good".*

Ceci s'explique aisément : chaque ensemble complet de choix doit impérativement contenir une assertion provenant de chaque disjonction. S'il se trouve que chacune des assertions d'une disjonction apparait dans un "no-good", il existe alors un environnement (l'union décrite précédemment), qui ne permettra pas de trouver une assertion de la disjonction de contrôle, pour donner un environnement complet et consistant : cet environnement là est donc contradictoire.

exemple : Soit la disjonction de contrôle suivante CONTROL*(X,Y)*, et les "no-good" déjà détectés : *nogood(A,X)* et *nogood(A,Y)*. La règle d'hyper-résolution détectera : *nogood(A)*.

2.4 Les raisons du choix d'ADDB

Dans le cadre de CIME, ADDB a été choisi pour gérer les hypothèses, lorsque les traitements le demandent. Les hypothèses, dans ce système, portent essentiellement sur le choix de méthodes à suivre et sur celui de leurs arguments, afin d'obtenir le résultat recherché.

Au premier abord, il peut sembler que la mise en oeuvre d'un tel mécanisme soit trop lourde, puisqu'il fait appel à des modules très complexes, tel qu'ATMS. Cependant, l'importance stratégique des choix pour la recherche de solutions, justifie que les moyens adéquats soient mis en place. D'autre par, cet algorithme, très général, a été adapté au cas particulier de CIME.2, où l'on ne retrouve pas tous les points exposés ci dessus (cf § 3).

Les autres raisons justifiant le choix de ADDB sont les suivantes :

- Sa souplesse qui permet de :

  1) rechercher une ou plusieurs solutions à un problème, sans qu'il y ait perte d'efficacité, comparativement aux autres méthodes. Dans la cas de CIME, ceci est appréciable car les différents sous-buts peuvent nécessiter des stratégies de recherche très différentes : on cherchera toutes les solutions quand il s'agira de *classer* la carte (on ne retient alors que la meilleure), ou une solution quand il s'agira d'en extraire des éléments caractéristiques, ou de focaliser la recherche.

  2) modifier dynamiquement la forme de l'espace de recherche, par ajouts ou retraits de disjonctions de contrôle. Ainsi dans CIME.2, pourra-t'on envisager de réduire l'espace de recherche au vu de certains résultats; faire appel à des procédures externes n'ayant pas toutes le même type, ni le même nombre d'arguments, sans avoir de gestion particulière à chaque cas.

- Ses performances

  Dûes au retour arrière chronologique qui permet de revenir sur le dernier choix effectué et non sur un choix quelconque (tel que cela se fait dans TMS, où seul le fait d'appartenir à l'environnement contradictoire, suffit pour qu'un choix soit remis en cause). Ceci évite les retraits "aveugles", et des tentatives inutiles.

Remarque : Il est possible que ces performances puissent être accrues du fait de l'utilisation de ATMS, car il permet une recherche efficace des "no-good" minimaux (ce qui devrait améliorer d'autant le retour arrière).

Il peut donc être intéressant, dans un système tel que CIME, où le nombre de données traitées est grand, et les traitements coûteux, d'expérimenter ces possibilités d'optimisations, et de mettre à profit les facilités offertes par cet algorithme.

3 C.H. : Une adaptation de A.D.D.B. à CIME.2

Nous avons signalé que, dans la nouvelle version de CIME, les hypothèses étaient faites suivant un principe s'inspirant de ADDB. Rappelons que dans CIME.2 ce sont les méthodes et leurs arguments qui constituent les hypothèses.

De ADDB, il a été retenu la gestion des disjonctions de contrôle qui permet :

=> d'effectuer toutes les combinaisons d'éléments pouvant être sélectionnés;

=> une évolution dynamique de l'espace de recherche, par ajout ou retrait de disjonctions de contrôle;

=> des retours arrière, sur les choix effectués, qui soient contrôlés;

=> d'être sûr d'avoir une solution se réalisant pour un ensemble de choix complet.

Par contre, un des aspects les plus importants de cet algorithme a été laissé de côté : l'utilisation de ATMS pour la vérification de la consistance de l'environnement choisi. Malgré les avantages de cette utilisation, qui ont été expliqués précédemment, ce module serait inutile dans le cas de CIME.2. En effet, la méthode choisie et ses arguments pris séparément, n'ont pas d'intérêt en tant qu'environnements. C'est l'ensemble (méthode,arguments) qu'il est intéressant de tester.

exemple : Dans un premier temps, CIME est utilisé pour trouver la meilleure chaîne de traitements permettant de classer, en termes de formes de végétales, une image satellitaire. Pour cela, plusieurs méthodes sont envisageables, avec des arguments pas toujours identiques. On pourra avoir les disjonctions de contrôle suivantes :

CONTROL(*methode1,methode2,methode3*)
CONTROL(*arg1,arg2,arg3,arg4,arg5*)
CONTROL(*P1,P2*)

où arg_i et P_j représentent des arguments pour les méthodes $methode_k$. Il se peut aussi qu'une méthode $methode_1$ nécessite d'autres arguments. On aura alors :

SI $methode_1$
ALORS *active*(CONTROL(*X,Y,Z*))
ET *active*(CONTROL(*A,B*))

Dans le cas où C.H. sélectionne la méthode $methode_1$, la pile des disjonctions de contrôle devient :

CONTROL(*X,Y,Z*)
CONTROL(*A,B*)
CONTROL(*arg1,arg2,arg3,arg4,arg5*)
CONTROL(*P1,P2*)

Inversement, une méthode $methode_p$ peut ne pas avoir besoin des arguments P_i :

SI $methode_p$
ALORS *passive*(CONTROL(*P1,P2*))

Dans ce cas, la pile, après sélection de la méthode $methode_p$, et application de la règle ci dessus, devient :

CONTROL(*X,Y,Z*)
CONTROL(*A,B*)
CONTROL(*arg1,arg2,arg3,arg4,arg5*)

Ces règles ne concernent que les disjonctions de contrôle, et sont gérées normalement par le moteur d'inférences, car elles ne se distinguent pas des autres règles de production.

B] GESTION DE LA NON MONOTONIE : LE CHOIX DE TMS

Comme nous l'avons signalé précédemment (c.f. chapitre I, B] § 2), il sera souvent nécessaire dans une session de CIME, d'avoir à enlever de la base de faits, des faits précédemment validés. Le retour sur un fait nécessite de la

minutie : en effet, il faut pouvoir maintenir les faits indépendants du fait enlevé, et remettre en cause les autres, ce qui nécessite par ailleurs une gestion adéquate des règles.

Le problème posé est donc celui de l'optimisation des mises à jour de la base de faits. C'est pour effectuer ces mises à jour que nous avons choisi TMS. Il existe d'autres façons de gérer la non monotonie : elles seront présentées dans un premier temps, puis le principe de TMS sera exposé.

1 Rappel sur la gestion de la non monotonie dans CIME.1

Dans la première version de CIME, il avait été nécessaire de dater chaque fait, afin de faciliter les mises à jour. Au niveau des règles, ces mises à jour étaient effectuées par appel d'une procédure dite interne (écrite en PROLOG). La grande complexité de la mise à jour dans ce cas, faisait qu'il avait été choisi d'enlever tous les faits validés après le fait enlevé, et par la suite, de valider à nouveau, les faits qui n'auraient pas du être enlevés. Pour cela, des règles étaient déclenchées plusieurs fois.

Cette gestion s'est révélée très coûteuse en temps. De plus, elle se mêle au contrôle si fortement qu'une certaine confusion en résulte. Il fallait donc que CIME.2 résolve autrement le problème de la gestion de la non monotonie.

2 Les différentes gestions de la non monotonie

2.1 La non monotonie dans SNARK

Avant d'exposer les différentes sortes de retraits que propose SNARK, il est important de parler de la syntaxe propre à cet environnement. Signalons tout d'abord qu'elle permet, en conclusion de règle, de nombreuses actions qui modifient implicitement ou explicitement la base de faits.

a) Les faits dans SNARK

Ce sont des triplets de mots, à priori sans signification. Dans un fait SNARK, le mot central désigne simplement une relation entre les mots l'encadrant. De façon générale, ces triplets peuvent être interprétés comme suit :

<objet> <relation> <valeur>
<nom1> <nom2> <nom3>

D'un fait à l'autre, un même mot peut occuper des positions différentes, et donc jouer des rôles différents. C'est pourquoi il a été dit, qu'à priori, ces mots n'avaient pas de signification.

exemple : l'assertion " *a est un élément de E* " se traduit par le triplet "*a element E*". On pourra aussi traduire l'assertion " *un élément de E est a* " par le triplet " *E element a* " sans qu'il y ait d'ambiguïté.

b) Les règles dans SNARK

Nous n'exposerons pas entièrement les possibilités offertes par le langage de SNARK pour écrire les règles. Nous présenterons seulement la syntaxe et quelques unes des facilités, proches de la programmation, qu'elle recouvre. Pour plus de détails voir [Laurierre88].

Les règles de SNARK sont des règles de production classiques intégrant la notion d'**antécédents** (ou conditions), et de **conséquents** (ou conclusions). Leur syntaxe est la suivante :

```
Règle <nom>                 *nom de la règle
  <entête>                  *SI,DESQUE,APRES
      <antécédent>+         *une ou plusieurs fois
  ALORS
      <conséquent>+         *une ou plusieurs fois
```

Les conséquents sont exécutés dans l'ordre où ils sont écrits, et peuvent faire appel à des opérations par l'intermédiaire de mots clé.

c) Les retraits de faits dans SNARK

Dans SNARK, le retour arrière par retrait d'assertion, se fait par le biais d'une commande : REVENIRSUR(fait). Cette commande de SNARK assure les mises à jour adéquates : toutes les assertions, déduites à partir du fait enlevé, sont retirées; toutes les assertions, enlevées à cause de lui, sont rétablies; et ceci récursivement.

Les retraits de faits dans SNARK, peuvent prendre d'autres aspects que le retour arrière : il y a possibilité de retrait implicite, et explicite. Le retrait implicite se fait lorsqu'il y a affectation : l'opérateur '<--' (objet <-- val), provoque la destruction de toutes les autres valeurs de l'objet, et lui affecte la seule valeur "val". Le retrait explicite d'information se fait par des commandes :

* **TUER <nom>** : supprime toutes les occurrences, relations, objet ou valeur de l'entité désignée;

* **TUERFAIT <nom1> <nom2>** : supprime toutes les occurrences du couple (objet,relation), représenté par (<nom1>,<nom2>);

* **TUERFAIT <objet> <relation> <valeur>** : supprime exactement le fait désigné.

Dans le cas qui nous intéresse, la commande REVENIRSUR serait la plus appropriée pour les mises à jour. Dans CIME.1, c'est ce que l'on a tenté d'implémenter. Il s'est révélé difficile de mettre en place une telle commande ("*action interne*" dans CIME.1), pour gérer aussi bien le retour arrière que les réinitialisations. Aussi avons-nous cherché une autre solution pour la nouvelle version de ce système.

2.2 La non monotonie dans les systèmes utilisant TMS

Les faits à enlever, sont communiqués au Truth-Maintenance System par le module déductif. L'originalité de la gestion du retrait de faits dans ces systèmes, réside dans deux caractéristiques :

a) Les faits à enlever ne sont pas retirés physiquement de la base de connaissances : ils changent seulement de statut, qui de **IN**, passe à **OUT**. Ceci est très appréciable, puisque cela évite, dans le cas où ils seraient validés une nouvelle fois par la suite, d'avoir à reconstruire les différentes données constituant un fait (notamment, la liste des justifiés). D'autre part, même s'ils ne sont pas, par la suite, validés de nouveau, le maintien de ces faits de statut **OUT**, permettra le cas échéant, de fournir des explications sur les raisons d'un échec de la recherche.

b) Le maintien de la cohérence de la base de fait, est assurée par le module TMS, suivant un algorithme de propagation de pointeurs. C'est ici que s'exprime toute l'importance de la structure de données choisie pour TMS (cf. **A]** 1.4). Dans ce cadre, les mises à jour, gérées par TMS, s'effectuent lorsque le statut d'un fait passe de **IN** à **OUT** par invalidation d'une assertion; ou, inversement, lorsque ce statut passe de **OUT** à **IN** par ajout de nouvelles justifications.

Algorithme de gestion des ajouts et des retraits de faits : D.B.G.C. (DataBase Garbage Collector)

Lors de ces mises à jour, les faits supportés sont examinés, leur liste de supporteurs et leur statut recalculés, et ceci récursivement. Les faits ajoutés sont transmis par le module déductif avec leurs justifications, les faits à enlever sont simplement signalés à TMS, et la mise à jour se fait comme suit :

* **Ajout de F et de <Liste de justifications>**
  Il y a mise à jour des justifiés de F, par introduction de F dans la liste des justifications.

* **Suppression de F**
  Pour chacun des justifiés F'de F

->Si F' n'est pas supporté par F :il y a mise à jour de ses justifications, par suppression de toutes celles contenant F.

->Si F' est supporté par F

-Le statut de F' passe de **OUT** à **0** afin de rechercher un autre supporteur pour F' (Le statut **0** est un statut **OUT** provisoire, qui ne sert que le temps de mise à jour).

-F' est mis provisoirement dans une liste L de faits à étudier.

-Ce même processus est appliqué récursivement aux supportés de F'.

Remarque : L'utilisation du statut provisoire 0 et de la liste d'examen L, sert à éviter les justifications circulaires.

exemple : F -> F'
F' -> G
G -> F'

Ce cycle s'arrête de deux façons :

1) Un élément de la liste L a son statut qui passe de **0** à **IN** : un autre supporteur lui a été trouvé. Dans ce cas, tous les éléments de liste sont réexaminés.

2) On est arrivé à des faits initiaux ou à des hypothèses sans qu'aucun élément de la liste ait eu son statut changé. Alors, F' est affecté d'un statut **OUT**, ainsi que tous ceux de la liste.

Ce que n'expose pas cet algorithme et qui est capital, c'est la marche à suivre afin de trouver, s'il existe, un autre suporteur à un fait de statut **0**. Cela revient à trouver une autre chaîne de justifications, partant du fait de statut **0**, et aboutissant à des faits sûrs ou à des hypothèses. Puisque tout fait de statut **IN** est correctement justifié, il suffit alors de rechercher dans la liste des justifications du fait examiné, une justification dont tous les éléments ont le statut désiré.

C] LE CHAINAGE MIXTE POUR LE MOTEUR D'INFERENCES (M.I.)

Tout comme dans la première version de CIME, le moteur d'inférences fonctionnera en chaînage mixte. L'intérêt de cette stratégie est qu'elle permet de focaliser la recherche sur le but en chaînage arrière (les éventuelles questions posées à l'utilisateur sont alors pertinentes), et d'obtenir les informations nécessaires par saturation en chaînage avant

(grâce au chaînage arrière, on ne déduit que ce dont on a besoin). L'espace de recherche est alors correctement maîtrisé, et la combinaison des recherches en largeur et en profondeur assure, en moyenne, un gain de temps dans l'aboutissement de la recherche.

Ce chaînage mixte se fait suivant un cycle : lorsqu'un but est fourni au système par l'utilisateur, le moteur d'inférences est invoqué en chaînage arrière pour déterminer les conditions à remplir pour satisfaire ce but. Dans cette démarche, il sera amené à satisfaire des sous-buts qui sont des étapes intermédiaires nécessaires à la satisfaction du but initial. Ces sous-but sont gérés par une pile de type LIFO (Last In First Out) : on parle alors de la pile des sous-buts.

Lorsque toutes les conditions sont remplies, le but est atteint. Il arrive généralement que certaines de ces conditions nécessitent l'interventions de l'utilisateur qui est le seul à pouvoir apporter une réponse pour satisfaire ou non ces conditions (La base de faits ne contenant pas les faits nécéssaires, ou ceux ci ne peuvent pas être déduits à partir des données de la base de connaissances). Dans ce cas, des questions lui sont posées, et les réponses enregistrées comme des fait nouveaux. Le chaînage avant est alors invoqué pour prendre en compte ces données nouvelles.

Le chaînage avant procède alors par **saturation** (c'est à dire qu'il prend en compte tous les nouveaux faits, et effectue toutes les inférences possibles). Lorsque plus aucune inférence ne peut être faite, le moteur retourne au chaînage arrière en mettant à jour la pile des sous-buts : certains d'entre eux ayant put être satisfaits par la validations de nouveaux faits pendant le chaînage avant, ces sous-buts sont alors dépilés.

Ce cycle s'arrête lorsque le but est atteint (la pile des sous-buts est vide), ou lorsqu'il n'y a plus aucune possibilité de l'atteindre.

CHAPITRE III

STRUCTURE DES PROGRAMMES-STRUCTURE DES DONNEES

A] STRUCTURE DES DONNEES

1 Présentation générale

Il faut que les connaissances mises en place permettent de façon pratique et explicite, une bonne gestion du système et le bon déroulement de l'inférence. Afin de pouvoir répondre à ces besoins, une structure de données adéquate doit être proposée. Elle doit favoriser la représentation des connaissances du thématicien et de l'informaticien, et ce avec le plus d'homogénéité possible, sans qu'aucune ambiguïté n'en résulte.

C'est dans cet esprit que nous avons mis en place une structure de données qui, par sa généralité, permet de représenter, sous le même formalisme, tout en les caractérisant, les données thématiques et informatiques

2 Structure des faits

2.1 Présentation de la structure de données

La gestion par TMS des ajouts et retraits dans la base de faits, impose une première structure qui, pour un fait F, est la suivante :

(F, LISTE DES JUSTIFICATIONS, LISTE DES JUSTIFIES, LISTES DES SUPPORTEURS, LISTE DES SUPPORTES, STATUT)

La signification et le rôle des éléments autres que F, ont été précisés dans le chapitre précédant (cf.chapitre II, **A]** § 1.4 et **B]** § 2.2), aussi insisterons nous surtout sur la syntaxe des éléments de type F.

Comme nous l'avons dit au chapitre I, dans CIME.2, on ne représentera pas seulement les pixels ou le contexte général, mais aussi des regroupements de pixels, une hiérarchie sur les différents éléments constituant l'image, ou sur les concepts, et éventuellement plusieurs images dites internes, nécessaires à la recherche.

exemple : Si l'on se propose d'étudier la déforestation dans la région où a été prise l'image, il faudra, dans un premier temps, *reconnaître* les différentes végétations de la région; en *dégager* les régions forestières et les régions qui les bordent; *établir et représenter* un lien hiérarchique entre ces zones de végétation et les pixels qui les composent; puis focaliser l'étude sur ces régions, afin de savoir s'il s'agit ou non d'un phénomène de déforestation.

2.2 Proposition d'une structure pour les faits

Donc, à un niveau plus élevé, se pose le problème de la représentation des différents éléments manipulés par le système. Cette représentation doit se faire au moyen d'une structure suffisamment générale, afin d'exprimer la connaissance de façon explicite et optimale, tout en évitant la multiplication des traitements particuliers. Il faut pouvoir travailler sur un élément et ses constituants (par exemple une zone et les pixels qui la constituent), sans qu'il y ait redondance des informations représentées. Pour cela, nous avons enrichi et étendu la représentation (**ATTRIBUT,OBJET,VALEUR**) utilisée dans CIME.1, afin de pouvoir représenter :

*Plusieurs éléments de nature différente, utilisés dans la recherche (les données de contrôle pour la gestion des hypothèses, les pixels, les zones, l'image elle même, les concepts, les liens entres ces concepts ...);

*Les liens hiérarchiques liant un élément à ses constituants, sous forme des propriétés que doivent vérifier ces constituants;

*Des propriétés sur certains descripteurs (ie. attributs) des éléments constituant l'image.

Pour parvenir à cela, la syntaxe des faits utilisés sera la suivante :

(TYPE de l'objet, NOM de l'objet, ATTRIBUT, LISTE)

où :

a) **TYPE** est un nom désignant la nature de l'objet. Les différents types sont mis en place par l'expert de façon initiale et factuelle, par le biais des faits initiaux présents dans la base de connaissances. De nouveaux types peuvent aussi être insérés dans le système par affectation dans les conclusion de règle. Cependant, on peut prédire l'existence de types tels que : **CONTROL** ou **PIXEL.**

b) **Nom de l'objet** est un nom désignant l'objet du type mentionné. Cette double caractérisation d'un objet permettra, pour plus de simplicité, que tous les objets soient numérotés, sans qu'il y ait d'ambiguïté.

c) **ATTRIBUT** est le nom d'un des attributs qui caractérisent le type de l'objet décrit (par exemple, il peut s'agir de la *classe* pour les pixels, ou de *l'ensoleillement* pour le contexte général)

d) **LISTE** peut avoir l'une des formes suivantes :

(V1,V2,V3,...) : les $\mathbf{V_i}$ étant les valeurs de l'attribut **ATTRIBUT** pour l'objet décrit.

exemple : (pixel,48,classe,(rhodo,sapin))

(P1,P2,P3,...) : les $\mathbf{P_i}$ représentant des propriétés que doivent vérifier les éléments constituant l'objet décrit. Ces $\mathbf{P_i}$ ont la structure suivante :

$(type_C, att_C, cmp_C, val_C)$ où :

=>$type_C$ est le type des éléments qui doivent vérifier la propriété;

=>att_C est l'attribut sur lequel porte la propriété;

=>cmp_C est un comparateur;

=>val_C est une valeur.

La propriété étant : les éléments de type $type_C$, ayant une valeur V pour l'attribut att_C, qui vérifie la relation $V\ cmp_C\ val_C$, sont ceux que l'on recherche.

exemple : (zone,22,constituants,((pixel,altitude,<,31),
(pixel,classe,=,foret)))

Ce fait signifiant que la *zone 22* est constituée de *pixels* dont *l'altitude* est inférieure à *31*, et la *classe* égale à *foret*.

L'intérêt de cette représentation, est qu'elle permet de connaître les objets construits non pas par l'énumération des éléments qui les composent, mais par les propriétés que doivent vérifier ces éléments (ce qui entraîne un gain de place en mémoire, et rend d'autant plus facile la construction de nouveaux objets). D'autre part, cette représentation à l'avantage d'être explicite pour l'expert, et facile à utiliser dans le système. En effet, il suffit d'utiliser ces propriétés comme des filtres ou comme des conditions de règles, pour avoir l'ensemble des éléments concernés.

(H1,H2,H3,...) : les H_i représentant des liens hiérarchiques entre l'objet décrit,et un élément dont il est un constituant (ces liens étant du type : "élément de").

Les H_I ont la structure suivante :

(T_p,N_p) où :

= T_p est le type du "père hiérarchique" de l'objet, c'est à dire l'élément dont l'objet est un constituant.

= N_p est le nom du "père".

exemple : (pixel,31,pere,((zone,22),(parcelle,3)))

Ce fait exprime **l'appartenance** du *pixel 31* à la *zone 22* et à la *parcelle 3*.

Cette représentation à l'avantage de permettre l'accès direct aux éléments du niveau supérieur de la hiérarchie.

2.3 Exemple d'utilisation

On pourrait avoir dans la base de faits, les faits suivants :

```
1 ) (general,contexte,ensoleillement_min,(14))
2 ) (general,contexte,ensoleillement_max,(22))
3 ) (general,contexte,classe_init,(rhodo,sapin,prairie,chêne))
4 ) (pixel,3,altitude,(31))
5 ) (pixel,3,pente,(0.15))
6 ) (pixel,3,pere,((zone,1),(parcelle,8)))
7 ) (pixel,4,pente,(0.2))
8 ) (pixel,4,altitude,(31))
9 ) (pixel,4,pere,((zone,1),(general,contexte)))
10 )(zone,1,constituants,((pixel,pente,<,0.3),
                          (pixel,altitude,=,31)))
```

Ces faits permettent d'exprimer les choses suivantes : pour l'ensemble de l'image, l'ensoleillement minimal est 14 (fait 1), l'ensoleillement maximal 22 (fait 2), et les classes de végétations envisagées sont le rhododendron, le sapin, la prairie, et le chêne (fait 3). Le pixel de numéro 3, désigne un point de la région qui est à une altitude de 31 (fait 4), et sur une pente de 0.15 (fait 5), tandis que pour le pixel de numéro 4, ces valeurs sont 31 et 0.2 (faits 8 et 7). On a, par ailleurs, exprimé l'appartenance du pixel 3 à la zone 1 et à la parcelle 8 (fait 6), celle du pixel 4 à la zone 1 (fait 9), ainsi que les propriétés des éléments constituant la zonel (fait 10) : ce sont les pixels pour lesquels l'altitude est 31,

et la pente inférieur à 0.3 (Ce que vérifient les pixels 3 et 4).

2.4 Les "cas particuliers"

2.4.1 Les attributs

La structure présentée est aussi utilisée pour décrire des éléments autres que ceux ayant un rapport direct avec l'image. Nous parlerons ici des **attributs** qui représentent les concepts que l'expert rattache aux éléments iconiques, et qui permettent d'interpréter l'image.

Tous les attributs ne sont pas concernés par cette remarque, mais ceux qui le sont, sont très importants, et touchent directement l'objet de la recherche. Il s'agit principalement des formes de végétation (ou ce qui a été appelé *classe* et qui à été présenté comme lien essentiel entre les entités iconiques et conceptuelles cf.chapitre I, § 2).

exemple : Pour illustrer ce propos, disons que si l'on traite de *forêt*, les régions plantées de *sapins* sont concernées,de même que celles plantées de *chênes* ou de *rhododendrons*.

C'est cette hiérarchie de concepts que l'on a aussi voulu exprimer, afin d'éviter à l'expert, au moment d'écrire les règles, d'avoir à décrire explicitement, et à chaque fois, les éléments qu'il considère. Il lui sera alors possible de rester à un certain niveau descriptif, proche de sa façon de raisonner.

Du point de vue syntaxique, il n'y a pas de changements comparativement aux autres faits : seule la signification des faits s'en trouve changée. Cela a l'avantage de permettre l'expression de ces particularités de la même façon que pour le reste de la connaissance. Un nouveau type est nécessaire : le type **ATTRIBUT**. La référence à une, ou aux valeurs, d'un attribut que l'on décrit, peut se faire par un "*attribut d'attribut*" : **VALEUR**. On pourra alors écrire le fait suivant :

(**ATTRIBUT**,classe,**VALEUR**,(prairie,rhodo,sapin,chêne))

Ainsi *classe* devient un objet que l'on peut décrire :

(**VALEUR**,forêt,pere,((**ATTRIBUT**,classe)))
(**VALEUR**,forêt,**VALEUR**,(sapin,rhodo,chêne))
(**VALEUR**,rhodo,pere,((**VALEUR**,forêt)))
(**VALEUR**,sapin,pere,((**VALEUR**,forêt)))

Ces faits indiquent que le type de l'objet forêt est **VALEUR**, et que son père dans la hiérarchie considérée est l'objet *classe* qui est du type **ATTRIBUT**. La valeur forêt est elle aussi décrite, puisque l'on sait les valeurs qu'elle peut prendre (sapin, rhodo, chêne). De même le lien hiérarchique ascendant permet de désigner l'ensemble (il s'agit ici de l'objet forêt) dont les objets *rhodo* et *sapin*, de type **VALEUR**, font partie. Il aurait été possible d'utiliser un attribut plus significatif pour exprimer ces relations, et d'écrire :

(**VALEUR**,rhodo,sorte_de,(forêt))
(**VALEUR**,sapin,sorte_de,(forêt))

Cependant, cette représentation fait que, pour comprendre, construire et utiliser la hiérarchie concernant ces objets, il deviendrait nécessaire de s'attacher aux sens des attributs qui les décrivent. On perdrait alors en généralité, puisque chaque relation ainsi décrite nécessiterait un traitement particulier de la part du système. Par contre, le biais syntaxique proposé ici pour représenter ce type de liens, demeure entièrement indépendant du sens des relations mises en jeu.

2.4.2 Les disjonctions de contrôle

Lors de la présentation de la méthode de gestion des hypothèses (c.f. chapitre II, **A]** § 2), les disjonctions de contrôle, gérées par une pile de type LIFO (Last In First Out), ont été notées :

CONTROL(*A,B,C,...*)

où *A,B,C,...* représentent des assertions incompatibles deux à deux (une assertion de chaque disjonction devant faire partie de l'environnement où se réalise la solution).

Les objets manipulés sont, comme nous venons de le voir, typés; d'autre part, la recherche d'un ensemble complet de choix nécessite l'intervention du moteur d'inférences, afin de modifier dynamiquement la pile des disjonctions, et de déterminer les inconsistances ou contradictions; enfin, TMS est utilisé pour le maintien de la cohérence de la base de faits lorsqu'il y a retour sur un choix ou réinitialisation.

Pour toutes ces raisons, les hypothèses émises ainsi que les disjonctions de contrôle doivent être représentées dans le formalisme des éléments manipulés par ces modules. Les disjonctions de contrôle seront donc écrites sous la forme de faits :

(CONTROL,SELECTION,CHAMP,LASS)

où :

CONTROL est le type de ces objets. On peut ainsi distinguer les disjonctions de contrôle des autres éléments présents dans la base de faits, tout en les utilisant de la même façon.

SELECTION est un stade de l'inférence, ou plutôt de la démarche suivie : par exemple, lors de la recherche d'une méthode de classification, **SELECTION** vaudra *"chaîne"* ou *"étape"*.

CHAMP peut être considéré comme un nom de disjonction. En effet, ce **CHAMP** donnera une indication sémantique sur les assertions contenues dans la disjonction de contrôle. En reprenant l'exemple précédant, **CHAMP** peut valoir : *"méthode"*, *"variable"*, ou *"parcelle d'entraînement"*.

LASS représente une liste de valeurs figurant les assertions de la disjonction de contrôle. Toujours avec le même exemple, cette liste pourra être : *(altitude,texture,radiométrie,pente)* pour le champ *"variable"*.

exemple : En reprenant plus complètement l'exemple de la classification de l'image, on aura les disjonctions de contrôle suivantes :

(CONTROL,chaine,methode,(dnp,sebest,salt))
(CONTROL,chaine,variable,(altitude,radiometrie,pente))
(CONTROL,chaine,parcelle_entrainement,(p1,p2))
(CONTROL,etape,methode,(dnp,sebest))
(CONTROL,etape,variable,(radiometrie,altitude,pente,texture))

Au cours de la recherche d'un ensemble de choix complet, les assertions choisies seront sous la même forme, mais avec un type différent : le type **ENVT**, pour environnement. La liste **LASS** ne contiendra alors qu'un seul élément :

(ENVT,chaine,methode,(dnp))
(ENVT,chaine,variable,(radiometrie))
(ENVT,chaine,parcelle_entrainement,(p1))

Après validation d'un ensemble complet d'hypothèses (consistant et pertinent suivant la base de faits à ce moment), ces choix seront maintenus dans la base de faits sous la même forme (c'est à dire que leur type ne sera pas changé), et seront utilisés pour l'inférence (ie. pour le déroulement d'une étape)

3 Représentation des règles

3.1 Présentation générale

L'expert devant écrire les règles, il est nécessaire qu'il puisse aisément faire référence aux informations contenues dans les faits, afin de pouvoir exprimer sa connaissance. La syntaxe des règles est donc soumise à deux conditions : être simple, et permettre d'exprimer la connaissance le plus complètement possible, et ce à plusieurs niveaux. Pour cela, la syntaxe proposée joue encore un grand rôle, et fait intervenir les outils disponibles dans CIME.2.

3.2 Syntaxe des règles dans CIME.2

Comme dans CIME.1, ce sont des règles de production de la forme :

SI
< *conditions* >
ALORS
< *conclusions* >

Ce qui change, comparativement à la première version, c'est l'utilisation de **mots clé**, de **fonctions propres** au système, et d'un **paramétrage** plus général. On retrouve les notions d'actions *internes* et *externes*, et celle de relation du type *attribut comparateur valeur*, en condition ou en conclusion.

3.2.1 Le paramétrage des règles.

Contrairement à ce qui se faisait dans la première version de CIME, il sera possible d'utiliser plusieurs variables dans la même règle. Lors de la compilation des règles, ne seront identifiées comme des variables, que les lettres précédées d'un point d'interrogation ?, aussi est-ce le format sous lequel elles devront être écrites.

Ces variables désigneront le nom des objets dont il sera question. Ce paramétrage implique donc, que l'application de chaque règle se fera tant qu'il sera possible de trouver des objets vérifiant les conditions. Le fait de pouvoir utiliser des variables différentes, permettra la référence à plusieurs objets de type différents, ou de créer de nouveaux objets, voire de nouveaux types. Cela implique que pour être référencé dans une règle, le type d'un objet devra obligatoirement être mentionné afin d'éviter toute ambiguïté.

3.2.2 Les mots clé

Il n'y en a que deux : **TYPE** et **ELEMENTS.** Le premier permet d'accéder au type d'un objet pour instanciation (en condition), ou pour affectation (en conclusion). On l'utilise simplement : **type ?x**, suffit à désigner le type de l'objet dont le nom est "**x**".

Le second mot clé, donne la possibilité de faire référence aux *"constituants"* d'un élément. Son utilisation, plus délicate, est expliquée dans le chapitre concernant les relations *"attribut comparateur valeur"* (cf § 2.3.4).

3.2.3 Les fonctions propres au système

La syntaxe des faits telle qu'elle a été présentée, ne permet pas de représenter toutes les informations dont on a besoin, et en particulier les relation spatiales existant entre les éléments iconiques (la proximité, le contact, la distance, l'inclusion ...) C'est pour palier ce manque que les **fonctions propres au système** ont été mises en place. Elles sont répertoriées dans un lexique, et connue du système comme des fonctions internes.

exemple : **contact ?x ?y** sera vraie si les entités iconiques désignées par **x** et **y** sont en contact. De même pourra-t-on avoir **inclus_dans ?x ?y**, ou encore **connexe ?x.**

Elle seront implémentées comme des fonctions booléennes, et pourront ainsi étendre la gamme des comparateurs utilisables en condition de règle. A ce titre, elle ne seront pas différentes des *Actions* : comme elles, elles seront lancées par le moteur d'inférences, leur résultat (vrai ou faux), équivalant à la validité ou l'invalidité d'une prémisse. Ces fonctions propres auront donc des arguments, qui seront des objets traités dans la règle où elles se trouvent.

3.2.4 Les relations "ATTRIBUT COMPARATEUR VALEUR"

Puisque les objets manipulés sont typés, et que les règles sont paramétrée par les noms des objets utilisés, les attributs devront eux aussi faire référence aux éléments auxquels ils se rapportent. En définitive, ces relations s'écriront :

attribut ?x comparateur valeur

où *attribut* désigne le nom de l'attribut dont il est question, *attribut ?x* désignant la valeur de l'attribut pour l'objet de nom "**x**". Comme dans la première version de CIME, on peut avoir pour *valeur* une valeur, ou une référence à la valeur d'un

attribut pour un autre objet (ou le même). Dans ce cas on aura :

$$attribut_x\ ?x\ comparateur\ attribut_y\ ?y$$

Dans ces deux cas, on suppose que les objets "x" et "y" ont déjà été identifiés par des relations **type ?x** et **type ?y**. Les comparateurs sont classiquement : l'égalité : "=", inférieur : "<" , supérieur : ">" , inférieur ou égal : "=<" , supérieur ou égal : ">=" , différent : "<>" (pour peu que le domaine des valeurs s'y prête). Nous avons signalé l'existence des fonctions propres qui, en premisse, palieront les limites de ces comparateurs (puisqu'en conclusion, seules les affectations sont autorisées).

exemple : si une règle ne doit être appliquée que sur deux objets de type zone qui sont en contact, on écrira :

```
SI type ?x = zone,
   type ?y = type ?x,
   contact ?x ?y,
Alors
   /conclusions/
```

Cette représentation s'enrichit encore de la possibilité de faire référence aux constituants des objets que l'on traite. Pour cela, on utilise le mot clé **elements**.

exemple : on voudrait sélectionner des éléments qui soient des zones de forêt dont les constituants ont une altitude inférieure à un certain seuil *s* (*s* représentant une valeur). On écrira :

```
Si type ?x = zone,
   classe ?x = forêt,
   altitude elements ?x < s
Alors
   /conclusions/
```

Ici, la condition porte directement sur une valeur d'attribut (l'attribut *classe*), pour des constituants. On pourra aussi faire appel à ce mot clé en combinaison avec l'autre mot clé **type** :

```
Si type ?x = parcelle,
   type elements ?x = zone,
   classe ?x <> classe elements ?x
Alors
   detruire ?x
```

Cette règle pouvant s'interpréter comme suit : *Si une parcelle est constituée de zones, et que ces zones n'ont pas la même classe que la parcelle, alors détruire cette parcelle.*

Dans ce dernier exemple, *detruire ?x* figure une action devant détruire de la base de faits tout ce qui concerne la parcelle "x" (Cette règle n'est pas issue du système, elle n'est là qu'à titre d'illustration).

De façon générale, toutes les combinaisons sont possibles. Cependant, deux restrictions sont à signaler :

1) Seules les combinaisons : **< type elements ?x >** et **<attribut elements ?x >** sont autorisées. Par exemple, on ne pourra pas écrire : **<attribut type ?x>**, ou encore **<type attribut ?x >**, etc ...

2) Les références hiérarchiques ne peuvent être faites qu'entre deux niveaux consécutifs de la hiérarchie décrite. Par exemple, on ne pourra pas écrire : **<attribut elements elements ?x>**, pour référencer les "*petits fils*" de "x".

Remarques : -Comme dans CIME.1, seule l'affectation est autorisée en conclusion de règle.

-Si une action doit produire de nouveaux faits, c'est l'interface qui se chargera d'insérer ces faits dans la base

B] STRUCTURE GENERALE DES PROGRAMMES DANS CIME.2

Le nouveau système comprendra plusieurs modules ou parties, chacun étant chargé d'assurer une des fonctionnalités du système. Ces modules sont reliés entre eux par des appels définis dans les sections suivantes. La structure générale de ce système provient de ses fonctionnalités d'une part, et des spécificités du domaine d'application et des recherches qu'il nécessite, d'autre part.

1 L'interface avec l'utilisateur

Cet interface est très important, puisque c'est lui qui est activé en permanence. Ce module d'interaction avec l'utilisateur permettra :

* La saisie, sous forme compilée ou non, des faits et règles constituant la base de connaissances, ou le chargement d'un fichier les contenant;

* D'effectuer des modifications dans la base de connaissances;

* La détermination d'un but, et le lancement de la session;

* En cours de session, de fournir des explications sur la recherche, ou de poser des questions à l'utilisateur (en chaînage arrière);

Ce module se présentera sous forme de menus, ce qui sera d'une utilisation agréable pour l'utilisateur, et il sera lancé dès l'appel au système.

2 Le gestionnaire d'hypothèses C.H.(Choix-Hypothèses)

CH, effectue des choix en fonction des données contenues dans la Base de Connaissances de Contrôle (B.C.C.). Chacun de ces choix est contrôlé, et éventuellement validé par le Moteur d'Inférences (M.I.) avec l'aide du Truth-Maintenance System (T.M.S.) et des règles de contrôle issues de B.C.C.. Rappelons que CH ne fournit que des ensembles complets de choix, aussi, fournira-t'il l'ensemble (méthode,arguments) nécessaire à une étape.

Lorsque l'utilisateur fournit au système un but, le moteur d'inférences, par application de règles de production, invoquera l'algorithme de choix de la même façon qu'une action. C'est donc l'expert qui, par le biais des règles, et de l'application des critères de satisfaction, qui demandera ou non la mise en place d'une autre étape. Il y aura deux façons de faire appel à cet algorithme : pour un choix donnant une étape de plus à la chaîne (la procédure appelée est **CH**), ou pour remettre en cause un choix (dans ce cas, la procédure appelée est **RETOUR-CH**). La distinction entre les deux procédures permet à l'algorithme de détecter les "no-good", et ainsi de gérer correctement l'émission d'hypothèses. Une fois que les choix ont été effectués, il sont mis sous forme de faits dans la base de connaissances. Le moteur reprend alors le contrôle, en partant en chaînage avant sur ces nouveaux faits. Au niveau de CH, lorsqu'il y a retour pour un autre choix, trois situations se présentent :

1) L'étape a été jugée bonne suivant les critères

CH procède alors à un nouveau choix, le met dans la base de connaissances, et retourne au contrôle qui lancera le moteur en chaînage avant.

2) L'étape n'a pas été jugée bonne suivant les critères

CH mémorise l'ensemble des étapes constituant la chaîne comme un "no-good". Il effectue un nouveau choix, pour l'étape ayant conduit à l'échec. Si ce choix n'est plus

possible, il retourne aux choix précédants (voir ADDB : chapitre II **A]** § 2). Ces nouvelles dispositions sont mises dans la base de faits, dont TMS assurera la mise à jour. Comme précédemment, le moteur est ensuite invoqué en chaînage avant au retour de l'appel.

3) Plus aucun choix n'est possible

C'est la fin de la session : CH en informe le moteur d'inférences qui terminera son travail et retournera à l'interface avec l'utilisateur, qui lui même, fournira les résultats de la recherche.

3 Le module du Moteur d'Inférences : M.I.

Il fonctionne en chaînage mixte à partir des données (règles et faits) de l'une ou l'autre des bases de connaissances qui, dans la réalité, ne sont pas distinctes. Il déduit des faits qui enrichiront la base de faits; il lance les actions contenues dans les règles et intègre leur résultats à la Base de Faits (B.F.), ces actions pouvant avoir pour conséquence le retrait de faits (T.M.S. se chargeant de rétablir la cohérence de la base de faits).

3.1 Les relations entre le MI et CH

Comme nous venons de le voir, c'est le moteur d'inférences qui appele le gestionnaire d'hypothèses C.H.. Cet appel se fait sans passage de paramètres, CH disposant des données nécessaires. Comme pour les autres actions, au retour de CH, les communications entre ces deux modules, se font par le biais de la base de faits, qui contiendra les choix que transmet CH au MI.

3.2 Les relations entre MI et TMS

Suivant les nouveaux faits qu'il déduit, les faits qu'il enlève, ou les contradictions qu'il détecte, le moteur d'inférences fait appel à TMS en lui fournissant : le nouveau fait et ses justifications; le fait enlevé; ou l'ensemble des faits ayant conduit à la contradiction. Ces appels se passent comme des appels à une procédure : au retour, le moteur d'inférences reprend la recherche avec la base de faits mise à jour.

3.3 Les relations entre MI et la Base de Connaissances BC

Ces relations sont très étroites, puisque le moteur d'inférences travaille avec les éléments de la base de connaissances. Il applique les règles qui y sont, en fonction des faits contenus dans la base. Par application de ces règles,

il ajoute ou enlève certains faits. Ceci peut aussi se faire par le biais d'actions internes déclenchées par le moteur. Dans le cas d'actions externes, ces communications avec BC se font par l'interface avec l'extérieur IE.

3.4 Les relations entre MI et l'Interface avec l'Extérieur IE

Si le moteur doit déclencher une action dite externe, il s'adresse à IE, l'interface avec l'exterieur. Il lui fournit le nom et les arguments des procédures à activer. Le module IE organise l'appel à la procédure, en mettant les données, fournies par le moteur d'inférences, sous la forme qui convient.

4 Le module d'Interface avec l'Extérieur IE

Il est un des modules importants du système puisqu'il permet de : mettre en correspondance l'image, qui est "extérieure" au système, et sa représentation interne,et de lancer des actions externes, puis d'intégrer leurs résultats au système. De ce fait, il ne communique qu'avec le moteur d'inférences, et la base de connaissances.

4.1 Les relations entre IE et le Moteur d'Inférences MI

Ces relations ont été décrites précédemment. Rappelons seulement que le moteur fait appel à IE lorsque des actions externes doivent être lancées, IE se chargeant d'effectuer correctement ces appels.

4.2 Les relations entre IE et la Base de Connaissances BC

Il faut signaler que, dans le cas d'une demande de mise en correspondance de l'image et de sa représentation, IE utilise des **primitives** qui effectuent la propagation des transformations d'une image dans l'autre. C'est dans ce cadre que se font les communications entre IE et la base de connaissances, qui comprennent aussi l'intégration dans la base de faits, des résultats des procédures externes. Ces primitives devront permettre de :

1) Réétiqueter une image;

2) Dégager, sous forme de propriétés, les nouveaux éléments de l'image traitée;

3) Utiliser des propriétés, afin de dégager de nouvelles entités iconiques constituant la carte;

4) Procéder à une propagation des propriétés d'éléments vers ceux qui les constituent, en s'aidant de liens hiérarchiques.

4.3 Les relations entre IE et l'extérieur

Ces relations se font tout simplement par lancement de procédures, qui sont des actions externes ou des primitives. Elles couvrent surtout la mise en forme des données transmises ou reçues.

5 Le module T.M.S.

C'est celui qui est chargé de maintenir la cohérence de la base de faits lorsqu'il y a ajouts ou retraits de faits. Son rôle dans le système a déjà été largement décrit , de même que ses relations avec les autres modules. Il est appelé par le moteur d'inférences chaque fois que celui-ci procède à un ajout ou un retrait de fait. Dans le premier cas, le moteur lui fournit le nouveau fait et ses justifications. Dans le second, seul le fait à enlever est transmis. Puisqu'il se charge du maintien de la cohérence de la base de faits, il intervient sur la Base de Connaissances BC en changeant le statut des faits, lorsque les mises à jour le demandent.

6 Les modules B.C et B.C.C.(Base de Connaissances et Base de Connaissances de Contrôle)

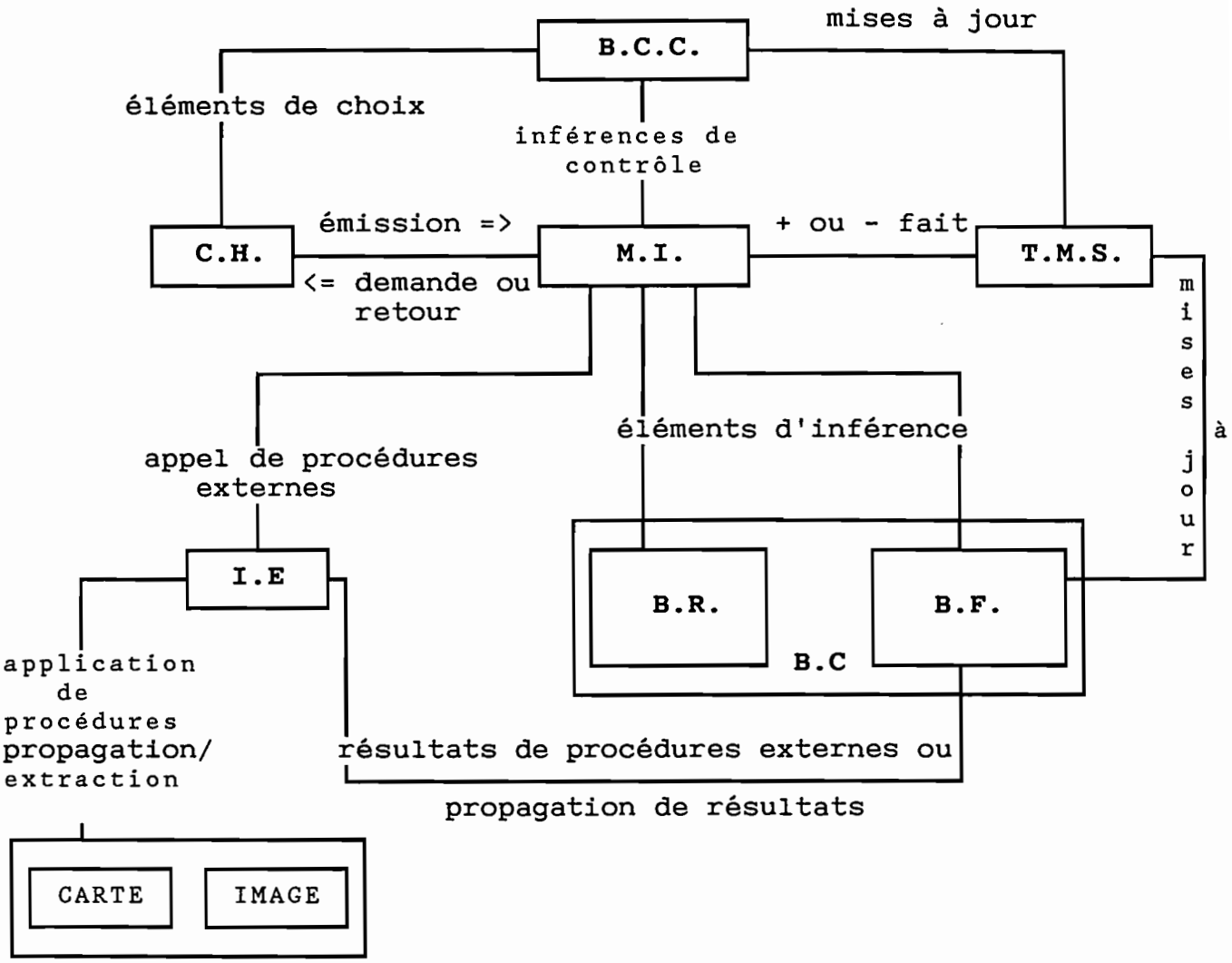
Rappelons que dans la réalité ils ne sont pas séparés : ils ne sont dissociés ici que pour mieux exposer le rôle des autres modules, et ce sur quoi ils interviennent. La base de connaissances comporte donc les données thématiques (BC) et des données de contrôle (BCC). Ces données sont de deux type : les faits et les règles. La base de connaissance est en relation avec l'ensemble des modules.

Remarque :

Comparativement à la première version de CIME l'architecture générale du système s'est enrichie des modules CH et TMS, ainsi que d'un interface avec "*l'exterieur*" plus sophistiqué qui permettra d'étendre les possibilités du système.

7 Schéma fonctionnel du système

Ce schéma a pour seule raison d'être, l'illustration de ce qui a été décrit précédemment. Les différents modules y sont représentés, ainsi que les relations qui les lient.


B.C.C.
mises à jour
éléments de choix
inférences de contrôle
émission =>
C.H.
<= demande ou retour
M.I.
+ ou - fait
T.M.S.
mises à jour
éléments d'inférence
appel de procédures externes
I.E
B.R.
B.F.
B.C
application de procédures propagation/ extraction
résultats de procédures externes ou
propagation de résultats
CARTE
IMAGE

LEXIQUE

Entité iconique : Ce terme désigne une entité représentée sur la carte et ayant une signification relative au **thème**. Dans le cas de CIME.1, ces entités sont des pixels, dans CIME.2 ce pourront être aussi des régions ou des objets généraux.

Objet : Dans ce système, le terme "objet" est employé dans son sens général pour désigner un "élément". Il ne se rapporte donc pas à un formalisme tel qu'on l'entend souvent en informatique.

Segments : C'est un ensemble de bornes sur les variables décrivant les pixels et qui est associé à un contenu (Ce contenu étant la liste des valeurs de l'attribut classe).

Stratégie complète : Dans une stratégie complète, si il existe une suite de dérivations menant à l'échec, alors il existe aussi une suite de dérivations satisfaisant les critères de la stratégie (Par exemple, la stratégie de parcour de l'arbre de recherche en profondeur est complète).

Taxonomie : Science des lois de classification des formes vivantes.

Thématique : C'est le domaine, ou l'expertise qui se rapporte à un **thème**. Dans le cas de CIME, la thèmatique se rapporte à la géographie, et plus particulièrement aux formes végétales.

Carte thématique : C'est une carte mettant en relief les différents éléments relevant du thème considéré.

Zones d'entraînement : Dans le cas de CIME, ce sont les ensembles de pixels, choisis parmi ceux qui constituent l'image, et utilisés pour produire les **segments**. Ce sont des paramètres des procédures externes, qui dans CIME.1 sont déjà fournies à la procédure (comme des données standard), et qui, dans CIME.2, pourront faire l'objet d'un choix.

BIBLIOGRAPHIE

[Avignon88] Mering C.,Blamont D.,Monjanel F., Ganascia J.G. : CIME : Une application des systèmes experts à la télédétection. Actes des 8eme rencontres internationalles d'Avignon "Les Systèmes experts et leurs applications";1988.

[deKleer86a] de Kleer J. : An Assumption-Based Truth-Maintenance System.Artificial Intelligence vol 28,1;1986

[deKleer86b] de Kleer J. and B.C. Williams : Back to Backtracking : Controlling the ATMS. Actes du colloque AAAI, vol 2,pp 132-139;1986.

[Doyle79] Doyle J. : A Truth-Maintenance System. Artificial Intelligence, vol 12,3;1979.

[Doyle83] Doyle J. : The Ins and Outs of reason Maintenance. International Conference on Artificial Intelligence,pp 87-90;1983.

[Laurière86] Laurière J.L. : Un langage déclaratif : SNARK. TSI, vol 5,3;1986.

[Laurière88] Laurière J.L. : Intelligence Artificielle,Tome 2,Représentation des Connaissances,ed EYROLLES,1988.

[Monjanel87] Monjanel F. : Rapport de stage,Système Expert Appliqué à la Télédétection,Université d'Orsay,1987.

[Shortliffe76] Shortliffe E.H. : Computer-Based medical consultations : MYCIN,American Elsevier;1976.

INDEX

Etifier Agnès.

Application des systèmes experts à la cartographie par télédétection. Bondy : ORSTOM, 1988, 60 p. multigr.

Mém. DESS : Informatique, Paris 11. 1988/09.