PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Développement d'un système expert pour l'évaluation des barrages en terre en opération
Auteur: Author:	Lionel Avon
Date:	1989
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Avon, L. (1989). Développement d'un système expert pour l'évaluation des barrages en terre en opération [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/57920/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/57920/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITE DE MONTREAL

DEVELOPPEMENT D'UN SYSTEME EXPERT POUR L'EVALUATION DES BARRAGES EN TERRE EN OPERATION

par

Lionel AVON

DEPARTEMENT DE GENIE CIVIL

ECOLE POLYTECHNIQUE

MEMOIRE PRESENTE EN VUE DE

L'OBTENTION DU GRADE DE

MAITRE ES SCIENCES APPLIQUEES (M. Sc. A.)

décembre 1989

© Lionel AVON 1989

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-58160-3

Canada

UNIVERSITE DE MONTREAL

ECOLE POLYTECHNIQUE

Ce mémoire intitulé :

DEVELOPPEMENT D'UN SYSTEME EXPERT POUR L'EVALUATION DES BARRAGES EN TERRE EN OPERATION

présenté par Lionel AVON

en vue de l'obtention du grade de MAITRE ES SCIENCES APPLIQUEES

a été dûment accepté par le jury d'examen constitué de :

M. Jean Lafleur, Ph. D., Président

M. Michel Soulié, D. Sc. A., Directeur de recherche

M. Claude Marche, D. Sc. A., Codirecteur

M. Jean-Charles Bernard, Ph. D., Membre

sommaire

Le but du projet de recherches présenté dans ce mémoire est la réalisation d'un système expert capable, à partir d'observations visuelles et instrumentales, de déterminer si il y a un mauvais fonctionnement dans un barrage en terre, puis, si tel est le cas, d'établir un ou plusieurs diagnostics possibles. Pour cela, il lui est indispensable de disposer d'un grand nombre de connaissances, et en particulier des connaissances des experts du domaine.

Un langage de représentation des connaissances et d'écriture de systèmes experts est développé sur le modèle des langages à objets. Ses caractéristiques en font un langage particulièrement adapté à notre problème.

Pour produire des faits et "raisonner" sur ces connaissances, un moteur d'inférences est écrit. Ce moteur est d'ordre 1, fonctionne en chaînage arrière, traite l'incertitude à l'aide de la théorie des possibilités, et peut utiliser des méta-connaissances.

Cette recherche aboutit donc à une ébauche de système expert, qui donne des résultats sur des problèmes concrets comme, par exemple, l'utilisation des critères de filtre, et à l'ouverture sur des perspectives et des directions de recherche de plus grande envergure.

abstract

The objective of the project presented in this thesis is to develop an expert system in the field of embankment dams. From visual observations and instrumental measures, the system will be able to detect and diagnose any malfunctions in any given dam. To perform this task, the system must be equipped with a great deal of knowledge, especially that of experts in the domain.

A representation language, in which we can also implement expert systems, has been developed based on the object oriented programming model. Its characteristics make this language particularly well adapted to our problem.

An inference engine has been written to produce new facts and to "reason". It is a first order engine which works by backwards chaining, which deals with uncertainty, and which can use meta-knowledge.

This research leads to the creation of an expert system that gives results on real problems, as for example the use of criteria for filters. It also widens our expectations of such a system and allows for new perspectives and directions of research in this field of study.

remerciements

Je tiens à remercier M. Michel Soulié d'avoir bien voulu accepter d'être mon directeur de recherche, et d'avoir permis que j'effectue ma maîtrise dans la plus grande autonomie.

Je remercie également le groupe du projet CastorPlus au sein duquel j'ai travaillé, et dont l'environnement et les infrastructures on facilité grandement le bon déroulement de ma maîtrise.

Je remercie le projet CastorPlus en particulier pour l'aide financière indispensable qu'il m'a accordée.

Je remercie enfin M. Jean Lafleur d'avoir accepté la présidence du jury, ainsi que M. Claude Marche et M. Jean-Charles Bernard d'en être membres.

table des matières

table des figures

introduction

Le présent mémoire expose les principales étapes et résultats du travail de recherche effectué pour cette Maîtrise en Sciences Appliquées à l'Ecole Polytechnique de Montréal, au département de Génie Civil, section Géotechnique, dans le cadre du projet CASTOR PLUS.

intelligence artificielle; systèmes experts

L'informatique est un outil efficace, indispensable depuis longtemps dans de nombreux domaines. Malgré le fait que certains problèmes courants demeuraient insolubles par les ordinateurs, on gardait bon espoir de voir naître, dans un avenir proche, des machines assez puissantes pour les résoudre avec les méthodes habituelles. Le problème était, croyait-on, un problème de puissance de calcul uniquement. On sait depuis quelques années que le principal problème est d'ordre méthodologique. Des chercheurs ont montré que même les ordinateurs les plus rapides imaginables ne résoudraient jamais certains problèmes avec les seules méthodes classiques d'alors. On a fait remarquer également que, si la vitesse de l'influx nerveux dans le cerveau humain est très lente comparée à celle de l'influx électrique dans un ordinateur (égale à la vitesse de la lumière), l'homme est encore --et de loin-- bien plus habile à résoudre certains problèmes que la machine.

La différence réside dans les méthodes utilisées : alors que l'homme raisonne, utilise son intuition et son expérience, et est conscient de ce qu'il fait, l'ordinateur exécute un algorithme qui, pendant la plus grande partie du temps, l'entraîne dans des calculs et des explorations inutiles. L'"intelligence artificielle" est née de ces constatations. Elle propose une nouvelle approche des problèmes par l'informatique; elle cherche --par définition-- à donner à la machine le comportement le moins "idiot" possible, i.e. lui faire éviter de perdre son temps. Elle cherche, car cela semble l'approche la plus raisonnable, à comprendre, modéliser et reproduire des comportements du cerveau humain. Les premières recherches ont dégagé plusieurs axes fondamentaux : représentation des connaissances, mécanismes de raisonnement (déduction, induction), auto-référence, etc... Des résultats théoriques et pratiques ont été produits; les principales réalisations effectives sont les "démonstrateurs de théorèmes" (dont Prolog est le plus connu), les systèmes à base de connaissances, les "systèmes experts", les systèmes de reconnaissance et compréhension des formes, des scènes, de la parole, des textes.

Les systèmes experts sont aujourd'hui, parmi ces réalisations, les plus utilisés dans des applications concrètes. Confinés à des domaines très restreints et très bien définis, ils utilisent la connaissance modélisée des experts du domaine en question, et produisent, pour des problèmes simples, des réponses semblables à celles des experts. Leur intérêt est multiple, outre le fait majeur que leurs méthodes leur permettent de résoudre des problèmes insolubles avec l'informatique classique. On peut, par exemple, leur donner la connaissance de *plusieurs* experts, et cette connaissance ne

disparaît pas avec l'expert. Ces avantages viennent s'ajouter à ceux apportés par le simple fait que l'on travaille avec une machine : infatigabilité, autonomie relative; on peut les reproduire à volonté et les utiliser en même temps à des endroits différents.

le projet de recherche : sujet, objectifs, réalisations

Le sujet du projet de recherche est le **développement d'un système expert pour l'évaluation des barrages en terre en opération.**

Le but ultime de ce travail est la réalisation d'un système expert capable, à partir des observations visuelles et instrumentales, de déterminer si il y a un mauvais fonctionnement dans un barrage, puis, si tel est le cas, d'établir un ou plusieurs diagnostics possibles. Pour cela, il lui est indispensable de disposer d'un grand nombre de connaissances de plusieurs types :

- connaissances générales des barrages et de leur conception,
- "façon de faire" des experts,
- données relatives à un site en particulier.

etc...

Le système est destiné à être utilisé sur les sites même; il doit donc fonctionner sur des ordinateurs autonomes et relativement compacts. On a choisi d'implanter la première réalisation sur PS2 d'I.B.M. avec le système d'exploitation OS2.

Les étapes à franchir sont --et tel est le plan de ce mémoire-- :

I. **représenter les connaissances.** Après avoir établi quels sont les besoins imposés par les objectifs du système expert et quelles connaissances lui sont nécessaires, nous décrirons une philosophie de représentation, un formalisme répondant à ces exigences : les "objets".

II. **se donner un langage de représentation.** Un langage basé sur le formalisme des objets est construit : RLA. Il combine des propriétés que l'on retrouve dans plusieurs langages à objets. Il a une très grande expressivité, des propriétés d'auto-référence, est interactif, et peut modéliser les divers types de connaissances indispensables à notre système. C'est un langage général de construction de systèmes experts.

III. **construire un moteur d'inférences** (i.e. un programme capable de déduire des nouvelles connaissances à partir de celles dont il dispose déjà, et de répondre à des questions). Après un rapide "état de l'art" de la recherche dans ce domaine, on optera pour un moteur original avec des caractéristiques encore inhabituelles : traitement de l'incertitude et utilisation de la méta-connaissance, fonctionnant dans un environnement d'objets. Un exemple complet d'utilisation sera donné, avant la présentation d'une approche pour la résolution de problèmes plus généraux dans le domaine des barrages.

Dans ce travail, le site du réservoir du Lac Ste. Anne est parfois utilisé comme exemple, car il présente des caractéristiques assez générales qui se retrouvent dans la plupart des autres barrages en terre, et suffisamment intéressantes pour que plusieurs experts aient déjà étés sollicités à son sujet. Il est important d'insister sur le fait que le système est conçu de façon

à pouvoir être utilisé sur un site *quelconque*. La plupart des données mêmes (connaissances relatives au génie civil, connaissances plus générales) pourront être utilisées *sans la moindre modification* pour des sites différents.

I. représentation des connaissances

A. objectifs et moyens du système

1. les experts

La conception, la construction, la maintenance et le suivi des barrages posent des problèmes très divers et englobent des domaines variés. Les tâches sont réparties dans plusieurs équipes qui interviennent les unes après les autres. Mais un barrage est une entité tellement complexe qu'il pose certains problèmes ne pouvant être résolus qu'avec une connaissance approfondie et globale. Les experts ont de telles connaissances, qui, combinées à leur expérience dans le domaine, leur permettent d'apporter des solutions ou simplement d'orienter les recherches.

Ainsi, par exemple, si le suivi et le contrôle du bon fonctionnement d'un barrage consistent d'abord à vérifier que toutes les mesures sont à l'intérieur d'intervalles prédéfinis, la détection d'une anomalie peut conduire à des recherches de causalité difficiles. Connaissant le cause de l'éventuelle anomalie, on pourra prendre les mesures de correction les meilleures. Un débit trop important dans un drain ou une charge trop faible ou trop forte dans un piézomètre sont des anomalies facilement détectables mais difficilement interprétables : leur cause peut être une fuite du noyau due à des fissures, elles mêmes dues à des tassements différentiels, ou un

mauvais fonctionnement d'un filtre aval ou amont dû au non respect de critères de conception, ou beaucoup d'autres choses...

L'explication que peut donner l'expert est vitale pour le barrage; elle permettra de prendre les mesures s'attaquant aux *causes* de la défectuosité, les seules permettant réellement au barrage de retrouver un fonctionnement normal. Cela peut être l'ajout de nouveaux éléments : drains, puits de décompression, tapis aval; de nouveaux appareils de mesure afin d'effectuer un contrôle plus précis; voire la définition d'une nouvelle politique d'exploitation du réservoir.

2. le système expert

Le but de notre système expert étant de produire des résultats comparables dans une certaine mesure à ceux des experts du domaine des barrages, il est légitime de chercher à lui procurer les mêmes éléments que ceux dont disposent les experts. Un inventaire des éléments intervenant dans l'expertise montre que ceux-ci sont tous d'un même genre : il s'agit de **connaissances**. Ces connaissances sont de types très différents :

- connaissances relatives au site étudié, cartographie, relevés des mesures et observations, historique de ces connaissances
- connaissances relatives aux barrages en terre, à leur conception, à leur construction, à la géotechnique, au génie civil en général
- connaissances plus générales en physique et en mathématiques
- méthodes de raisonnement propres au domaine, ou plus générales : déduction logique, induction, généralisation

- expérience acquise, et méthodes d'apprentissage (le plus souvent inconscientes), intuition

etc...

Il est indispensable que le système expert dispose, sous une forme ou sous une autre, de la plupart de ces connaissances, sans quoi il serait illusoire de prétendre pouvoir obtenir des résultats intéressants. On s'attachera donc d'abord à chercher un moyen de modéliser ces connaissances de façon à les rendre accessibles à un programme d'ordinateur. Pour cela, un langage de représentation sera spécialement conçu : RLA. Ce langage basé sur les "objets" sera très expressif, général et modulaire, et aura des caractéristiques particulières qui en feront un langage adapté à la construction de systèmes experts.

B. connaissances nécessaires

L'inventaire des connaissances mises en oeuvre par l'expert pendant son travail est la première étape de la réalisation de notre système expert.

1. données relatives aux sites

Les connaissances dont on semble avoir besoin en premier lieu sont celles relatives au site. Connaissances *factuelles*, ce sont elles que l'on sait généralement le mieux manipuler en informatique. Elles sont souvent quantifiées, chiffrées, et leurs domaines de valeurs sont connus; parfois, elles ne sont pas quantifiables, mais seulement descriptibles au moyen d'un petit nombre de symboles.

a) "cartographie"

On entend par "cartographie", l'ensemble des données descriptives des différents éléments, concernant leurs positions, leurs formes et leurs natures. Les éléments considérés sont toutes les parties du barrage, les fondations, le réservoir, les instruments...

Considérons l'exemple du réservoir du Lac Ste. Anne. Une vue générale de l'ouvrage est donnée en figure I B 1 b 1, ainsi qu'une coupe de la digue, en figure I B 1 b 2.

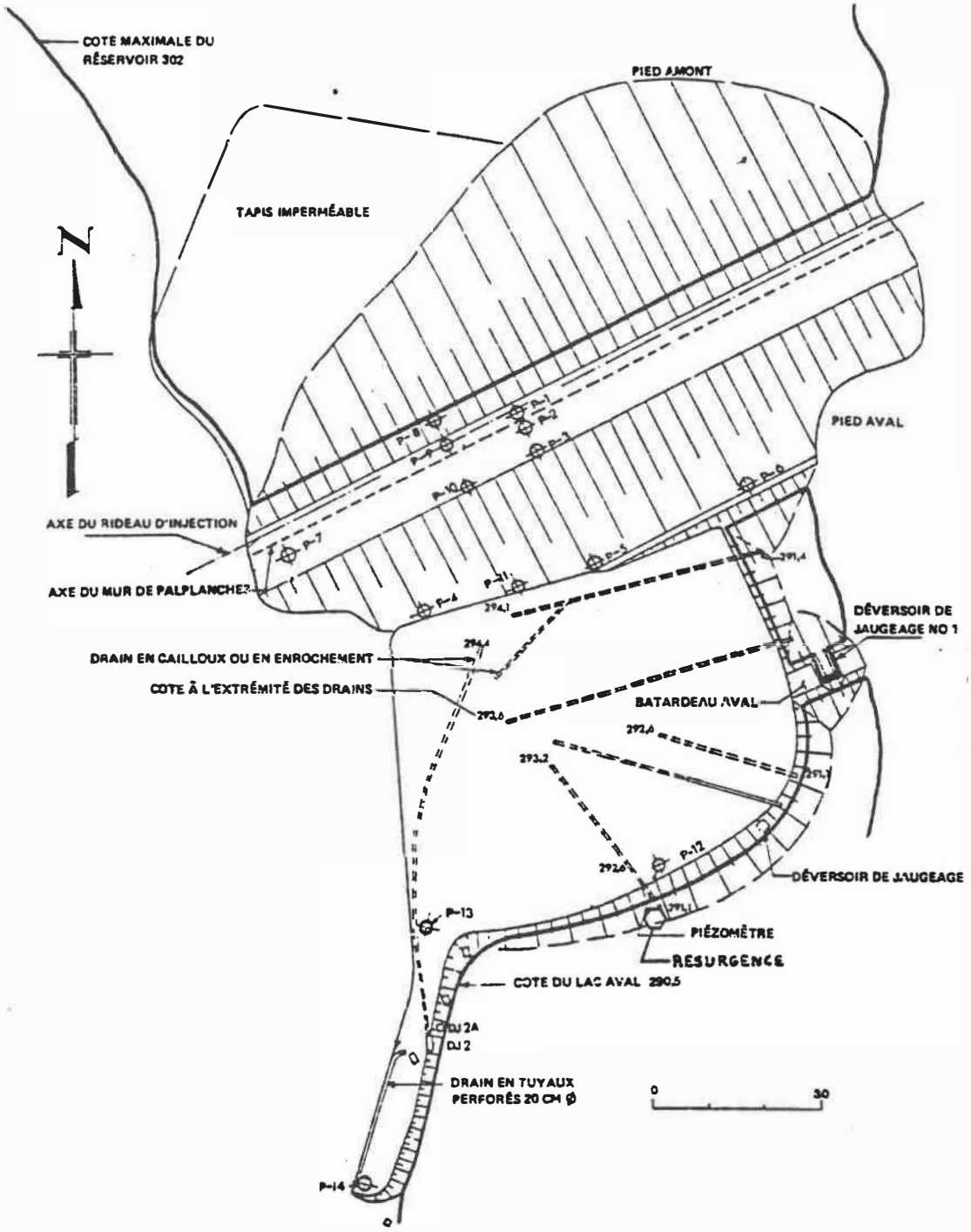

figure I B 1 b 1 : vue en plan de la digue du réservoir du Lac Ste. Anne (d'après [HYDRO 84], figure 1A)

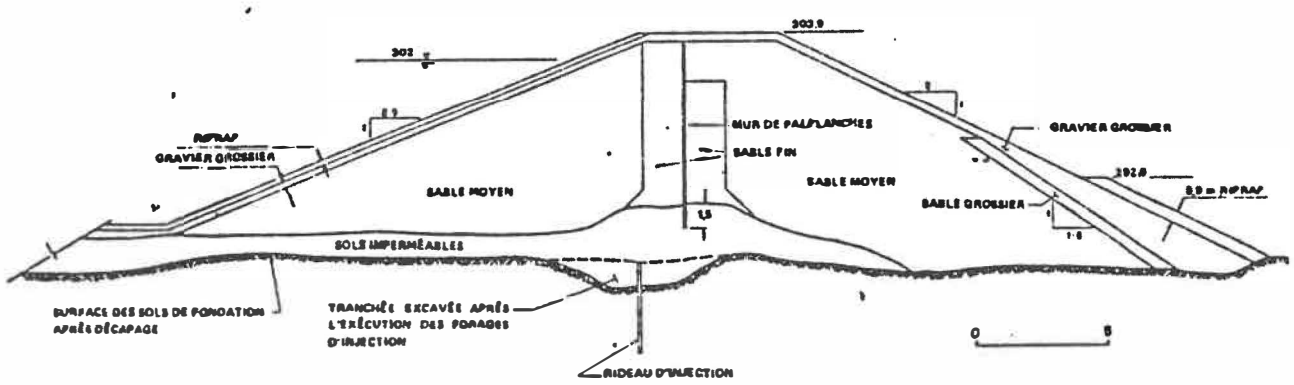

figure I B 1 b 2 : vue en coupe de la digue du réservoir du Lac Ste. Anne (d'après [HYDRO 86], figure 2)

La "cartographie" doit faire l'inventaire des pièces, avec une degré de précision qui dépend du type de l'expertise que l'on désire obtenir. Il est inutile, par exemple si l'on s'attache seulement à vérifier les critères de filtres, de donner les dimensions des objets au centimètre près, ou la granulométrie avec le pourcentage passant de chaque tamis standard. Seules les positions relatives des éléments ainsi que, conformément à la pratique courante actuelle, le pourcentage passant un ou deux tamis seront nécessaires.

b) appareillage et mesures

Les mesures effectuées par les appareils en place dans un ouvrage de retenue sont des éléments fondamentaux dans la connaissance que l'on peut avoir de l'ouvrage. Il rendent accessibles des paramètres importants pour le suivi, la détection des anomalies et la compréhension des

comportements. Les niveaux, charges et débits d'eau, les tassements, déplacements, déformations et autres, sont matérialisés par des résultats de mesures. Ceux-ci sont régulièrement collectés, contrôlés, étudiés, puis complètent des statistiques et des graphiques.

Les mesures sont facilement représentables en machine : elles se mettent toutes sous forme de nombres, et on en connaît bien les domaines de variations et les limites.

Des connaissances sur les appareils de mesure sont également nécessaires, en particulier leur précision, leur état et leur fiabilité, pour permettre d'établir la confiance que l'on peut accorder aux mesures, et pondérer chacune par une "incertitude".

c) observations

Si plusieurs ouvrages en terre ne sont pas instrumentés (comme par exemple dans la région de La Grande), tous font l'objet d'inspections. Les données d'observations visuelles sont indispensables au suivi du comportement des barrages. Elles sont aussi, en cas d'incident, les premières données que l'expert peut vouloir consulter.

La quantité de données produite par les inspections est immense. La nécessité de rendre leur accès facile et de les uniformiser a conduit des ingénieurs à chercher à les informatiser. Les travaux de Smith [SMITH 87] ont abouti à un formalisme de représentation des observations sous forme d'un "vocabulaire standardisé" et à un programme informatique capable de

les gérer. Les données informatisées sont centralisées et accessibles aux personnes autorisées.

Il est clair qu'un tel travail est de première utilité pour notre système expert. Le système expert devrait pouvoir y accéder automatiquement; il doit être, au même titre que les experts, une "personne autorisée". Avec des moyens simples de communication, les données pourraient circuler entre les deux systèmes sans aucun traitement, conversion ou codage supplémentaire, charge au système expert de les interpréter correctement.

d) historique

Le programme développé par Smith gère l'historique des observations des sites. Toutes les données acquises depuis le début de la mise en oeuvre du système sont accessibles au même moment. Ainsi peut-on étudier *l'évolution* d'un aspect particulier au cours du temps, et dégager les tendances. Par exemple, une augmentation du nombre de fissures en crête d'un barrage, événement crucial pour l'analyse du comportement, doit être facilement repérable et mise en évidence.

Ces notions d'évolution et de tendances sont au moins aussi importantes en ce qui concerne les résultats des mesures. Dans ce cas, les données étant numériques, le traitement apparaît plus aisé. On peut facilement quantifier les variations à l'aide du calcul approché des dérivées, même si les intervalles de temps des mesures sont très irréguliers (par exemple de quelques heures à quelques mois ou années).

2. connaissances relatives au génie civil

Les données relatives aux sites ne sont interprétables que dans la mesure où l'on dispose des connaissances relatives au domaine des ouvrages en terre. Ces connaissances, théoriques et livresques, ou pratiques et basées sur l'expérience, celles que possède tout expert du domaine à des degrés divers, sont aussi nécessaires au système expert.

Ainsi, des lois physiques de base en hydraulique et en géotechnique devrons être connues du système expert. Citons par exemple les propriétés de caractérisation des sols, leurs différents systèmes de classification, leurs propriétés structurelles, les effets de l'eau, les lois de l'écoulement, les lois de consolidation, les critères et mécanismes de rupture, etc...

Les connaissances qu'acquièrent les experts tout au long de leur pratique dans leur domaine sont également nécessaires au système expert. On trouve ces connaissances soit dans les publications des experts, soit directement auprès d'eux, en les interrogeant.

3. connaissances plus générales

Beaucoup de connaissances sont utilisées par l'homme automatiquement et de façon naturelle, sans qu'il en ait toujours conscience. Ces connaissances sont cependant primordiales pour ses raisonnements. Pourquoi croire que la machine pourrait s'en passer, si l'on espère d'elle des résultats comparables aux résultats humains?

a) mathématiques et logique de base

Le calcul a été la caractéristique fondatrice de l'informatique; il y est donc assez bien représenté (même si de nombreux progrès sont encore souhaitables, en particulier en ce qui concerne le calcul symbolique). A peu près tous les problèmes d'ingénierie font appel à des connaissances de calcul. Un système expert, même basé sur une représentation symbolique et "d'un haut niveau", se doit donc de pouvoir effectuer toute une panoplie de calculs, tant élémentaires que complexes.

Ces connaissances du calcul qu'a tout ingénieur s'accompagnent de connaissances des principes de la logique mathématique. La logique mathématique, même si elle n'est qu'une petite partie de ce que l'on utilise généralement pour raisonner, est à la base de plusieurs principes simples considérés comme fondamentaux dans le raisonnement. Il s'agit par exemple du principe du tiers exclu (une chose et son contraire ne sont pas possibles en même temps), du Modus Ponens (si A implique B, sachant A on peut déduire B), de la disjonction des cas (si A implique B, et A' implique B, alors (A ou A') implique B) etc... Bien que l'on utilise la plupart du temps ces lois sans s'en apercevoir, il est nécessaire de les rendre accessibles au système expert d'une façon ou d'une autre.

b) savoir-faire et connaissances inconscientes

Comment utiliser efficacement ses connaissances? C'est une question que l'on se pose rarement, mais à laquelle on répond implicitement à chaque fois que l'on raisonne. L'intelligence n'est pas fonction seulement de la masse des connaissances factuelles ou concernant des règles d'un

domaine particulier. Elle tient en grande partie aussi à l'habileté avec laquelle on retrouve et utilise les connaissances pertinentes pour le problème traité. Cette habileté, ce savoir-faire, sont des formes de connaissance le plus souvent inconscientes.

Un certain savoir-faire est traditionnellement représenté en informatique : les algorithmes. Ce sont des méthodes de calcul, de manipulation et de traitement des données.

Le "savoir-raisonner" est étudié depuis l'aube des mathématiques; plusieurs systèmes, tels la logique, formalisent bien certaines méthodes de raisonnement humains, comme on l'a vu précédemment.

Cependant la plus grande partie du savoir-faire est difficilement accessible, et encore plus difficilement représentable dans un formalisme de machine.

4. extraction des connaissances

Nous avons maintenant conscience de la multiplicité des connaissances mises en oeuvre dans la résolution de problèmes par un expert humain, en particulier dans le domaine des barrages.

Certaines de ces connaissances, mêmes celles spécifiques aux experts, sont assez bien connues et décrites. Les publications faites par les experts expriment clairement une partie de leur savoir, de leur expertise. Sherard [SHERARD 85], par exemple, donne des règles très précises sur la conception des filtres d'après son expérience. Ces règles, on le verra plus loin, seront facilement transmises au système expert.

Cependant beaucoup des connaissances sont difficilement exprimables, en particulier, par nature, celles utilisées inconsciemment. De nombreuses études psychologiques proposent des modèles pour rendre compte et tenter de répertorier quelques uns des mécanismes de raisonnement et autres connaissances. Détienne [DETIENNE 88] passe en revue plusieurs de ces études et donne des éléments d'analyse. Son travail porte sur "la compréhension des programmes informatiques" mais est assez général pour être bénéfique à la réalisation du système expert. Il s'inscrit en effet dans le cadre de la psychologie cognitive et aborde des caractéristiques de l'activité des experts. La théorie des schémas, modèle de représentation de connaissances, est son environnement de base. Cette théorie sera pour nous un point de référence dans la détermination des caractéristiques du langage dont nous voulons nous doter.

C. les objets : une "philosophie" de représentation

La théorie des schémas rend assez bien compte des comportements cognitifs mis en oeuvre dans l'activité des experts. Cette théorie est à rapprocher des notions de "cadres" (frames), "scénarios" (scripts), prototypage de concepts, héritage, instanciation, etc., autant de notions qui vont s'articuler autour des "objets" dans le langage de représentation dont nous avons besoin.

1. ce que l'on entend par "représentation"

Il importe ici de préciser le sens que l'on accorde au mot "représentation", afin de bien saisir la puissance et aussi les limites des différents langages informatiques de représentation.

a) représentation

D'après le dictionnaire, la **représentation** est le fait de *rendre sensible des objets ou concepts au moyen de signes.* En informatique, on cherche à rendre sensible les éléments (faits et méthodes) d'un domaine particulier au moyen de bits, d'octets, de nombres, de caractères, de symboles, de structures, de fichiers, de programmes, de bases de données, de bases de connaissances etc... Les langages informatiques sont des outils qui permettent de rendre effective la représentation.

Ainsi par exemple, les ordinateurs ont tous la possibilité de représenter les nombres et de représenter des méthodes qui, agissant sur eux, réalisent des opérations mathématiques. `1` est la représentation de l'unité; le résultat de `sin(1)` est la représentation du réel sinus de l'unité; sin, la fonction sinus d'un langage tel Pascal ou C, est la représentation d'une méthode numérique.

Le but du système expert est de représenter le plus possible des connaissances des experts du domaine, avec le maximum de précision et de fidélité.

b) syntaxe et sémantique

L'ensemble des règles qui régit les relations entre les formes de la représentation est appelé la **syntaxe**. Tout langage informatique a une syntaxe. Elle distingue précisément ce qui peut être écrit de ce qui ne le peut pas. Une donnée ou un programme *syntaxiquement* correct est accepté par l'ordinateur; cela ne veut pas dire qu'il représente nécessairement quelque chose qui a un sens!

La **sémantique** est ce qui consiste à attribuer un sens aux formes syntaxiquement correctes. L'intérêt d'une représentation réside dans sa sémantique. Ce sont des considérations sémantiques qui font en sorte que la représentation soit cohérente, que, si `sin` représente la fonction sinus, `sin(1)` représente le sinus de ce qui est représenté par `1`.

Un langage informatique est donc, d'un certain point de vue, un couple syntaxe-sémantique, dont la cohérence fait qu'on lui trouve un intérêt.

c) représenter des représentations

Des domaines différents et à priori quelconques peuvent être représentés, suivant la puissance d'expression du langage utilisé. Le domaine de "la représentation au moyen d'un langage informatique" peut donc, lui aussi, être représenté au moyen d'un langage informatique, à condition de disposer d'un langage assez expressif. Cette propriété de pouvoir représenter des langages de représentation confère aux langages qui la possèdent une puissance nouvelle. Une branche de l'intelligence

artificielle développe depuis quelques années de tels langages appelés RL (Representation Language), RLL (Representation Language Language)...

Un langage de représentation de langages est appelé "méta-langage". L'auto-référence est la propriété qu'a un méta-langage de se représenter lui-même.

2. les objets : "granules" élémentaires

La recherche d'un moyen de représentation le plus général possible a fait ressortir la nécessité de disposer d'unités élémentaires à la base de tout le système. Ces unités élémentaires, ces "granules", sont appelées les "objets". Des objets simples pourront être combinés entre eux et former des objets de plus en plus complexes et sophistiqués; des objets semblables se regrouperont dans des cadres (ou "frames"); des relations entre eux pourront matérialiser leurs interactions.

A la manière d'un édifice, le système expert sera donc construit à partir d'objets de base, combinés et reliés. Chaque objet conservera une plus ou moins grande autonomie vis-à-vis des autres objets, et pourra être considéré seul ou par ses relations avec les autres. La modularité est un des intérêts principaux de cette forme de représentation.

Tout élément du système sera d'abord un objet, quelque soit sa nature ou sa complexité. Tout ce que l'on veut représenter doit prendre la forme d'un objet. Ainsi les relevés de mesures, les rapports d'observations visuelles, mais aussi les connaissances en géotechnique et en hydraulique, l'expérience et le "savoir-faire" des experts, et les méthodes de

raisonnement même, devront être représentés par des objets. Un nombre, un symbole, une liste de propriétés, une matrice, un programme, une règle, une base de données, sont tous représentés par des objets dans le langage.

3. des relations entre les objets

Les relations qui existent entre les objets sont souvent aussi importantes, voire plus importantes, que les objets eux-mêmes. Par exemple le lien entre un fait et la règle qui a permis sa déduction est fondamental pour la compréhension du fait. Les relations entre les objets d'un cadre et le cadre (lui-même un objet), l'appartenance de plusieurs données à l'historique d'un même site, ou encore le lien qui exprime qu'une méthode est le cas particulier d'une méthode plus générale, sont autant de relations fondamentales pour l'organisation et la cohérence du système.

La gestion de toutes les relations entre les objets est un point clé du langage.

4. "objet = { données + procédures }"

L'équation objet = { données + procédures }, tirée de [ROCHE 89], synthétise un autre principe fondamental des langages objets. La programmation classique fait une distinction nette entre, d'un coté, les programmes, et de l'autre, les données, comme si ces notions étaient indépendantes. Or elles ne le sont pas : un programme est écrit pour fonctionner avec un certain type de données et pas un autre...

Un nouveau concept est proposé avec les objets : les données et les procédures qui les manipulent y sont regroupés. Les données, qui

définissent l'état d'un objet, ne peuvent être utilisées que par l'intermédiaire de procédures attachées à l'objet. L'unité sémantique qui existe entre un programme et ses données est ici réalisée syntaxiquement.

5. un système général de représentation

Cette "philosophie" de représentation basée sur les objets et leurs relations confère aux langages une grande expressivité; elle les rend très généraux et leur permet de répondre aux exigences des systèmes experts : représenter une grande diversité de connaissances. Les langages basés sur ce principe sont appelés "langages à objets" (ou "langages objets", ou "langages orientés objets").

II. RLA : un langage

Des recherches très récentes en intelligence artificielle ont abouti à l'écriture de plusieurs langages à objets. Ces langages ont beaucoup de caractéristiques communes, mais développent chacun des particularités propres. Le Laboratoire d'Informatique de l'Université de Caen est à l'origine du langage utilisé ici. Baptisé AIRELLE (Cf. [LIUC 88a] et [LIUC 88b]), ce langage s'est transformé avec les besoins (Cf. en particulier [AVON 88]); la version actuelle, appelée RLA, répond à de nouvelles exigences et est plus rigoureuse dans sa définition. AIRELLE et RLA sont maintenant deux langages distincts, basés sur les mêmes principes. Comme d'autres l'on fait (Cf. par exemple [ROCHE 89]), RLA tente d'être à la fois un langage de représentation et un langage d'écriture de moteur d'inférences, deux caractéristiques essentielles des systèmes experts.

Notons ici qu'écrit dans l'environnement Lisp [INRIA 87], RLA ne cache pas Lisp, dont la puissance de l'interprète et du compilateur reste totalement disponible.

A. un langage à objets

La méthodologie adoptée pour la construction du langage RLA est de définir les objets du plus simple au plus complexe, en enrichissant sans cesse l'ensemble. On appelle ce principe la **factorisation** : toute création

dans le langage s'efforce d'abord de tirer partie de ce qui existe déjà, en composant les méthodes, en combinant les structures d'objets, en ajoutant des "facteurs" aux termes existants.

1. types et héritage

Tout dans RLA est objet. La forme prédominante d'action sur les objets est l'**envoi de message**, qui obéit à la syntaxe suivante :

[*objet sélecteur paramètres*]

où

- *objet* est l'objet que l'on veut faire agir
- *sélecteur* est un nom d'action, ou **méthode**
- *paramètres* est une liste éventuellement vide des paramètres de l'action

Tout objet a un **type**. Un type caractérise un ensemble d'objets susceptibles de répondre aux mêmes messages, et regroupe les méthodes (i.e. les programmes) associées à ces messages. Cette notion de type englobe la notion habituelle de type dans les langages informatiques. On retrouvera en particulier tous les types courants : types de nombres, chaînes de caractères, types structurés divers.

Les types sont définis les uns par rapport aux autres au moyen d'une hiérarchie appelée **héritage**. L'héritage initial dans RLA est simple : un type ne peut hériter que d'un seul autre, i.e. un type est la spécialisation d'un seul autre. Le type `objet` n'hérite d'aucun type. L'héritage de RLA est

Page 25 manquante

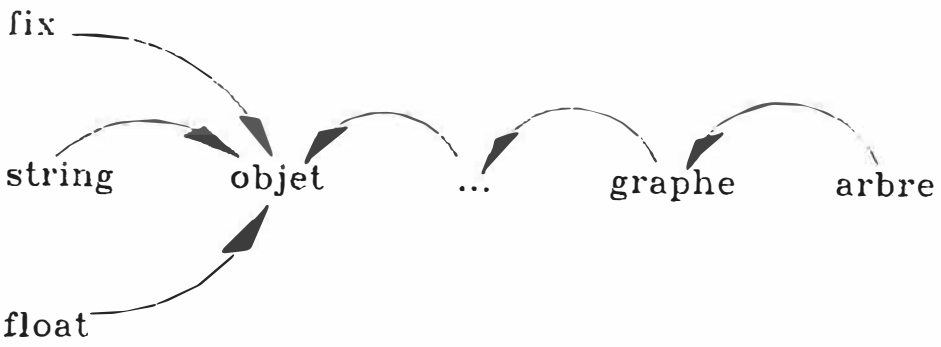

figure I D 1 a : une partie du graphe d'héritage

Le graphe d'héritage représente le graphe de la relation "hérite de" sur l'ensemble des types. Une flèche (sans talon) d'un type T+ vers un type T signifie : "T+ hérite directement de T", ou "T+ est une spécialisation de T", ou "tous les T+ sont des T". La relation d'héritage est transitive; ainsi, tous les types héritent du type `objet`. Ici on lit, par exemple, que tous les arbres sont des graphes, tous les graphes sont des objets, tous les arbres sont des objets, tous les entiers (`fix`) sont des objets...

2. classes, unités et instanciation

Les seuls types simples ne suffisent pas comme éléments de base au langage. D'une façon apparentée à celle des langages classiques tels Pascal ou C, RLA permet de définir des types structurés : les **classes**. Une classe définit des composantes, ou **attributs**, que ses objets pourront posséder. Il est important de noter que ces attributs ne sont pas statiques. Plusieurs objets d'une même classe peuvent avoir des attributs différents, et peuvent en posséder de nouveaux ou en "oublier" au cours de leur existence.

Ces objets structurés s'appellent des **unités**. Toute unité est créée par une classe, est l'**instance** d'une classe. Les unités ont des noms symboliques. On peut accéder à leurs composantes (les valeurs de leurs attributs) grâce à des envois de messages particuliers. L'unité est l'élément fondamental des langages objets; c'en est l'unité de base!

L'ensemble des relations entre les classes et leurs unités peut se représenter par l'**arbre d'instanciation** (Cf. figure I D 1 b).

La classe la plus générale, i.e. celle dont toutes les autres classes héritent, s'appelle `unité`.

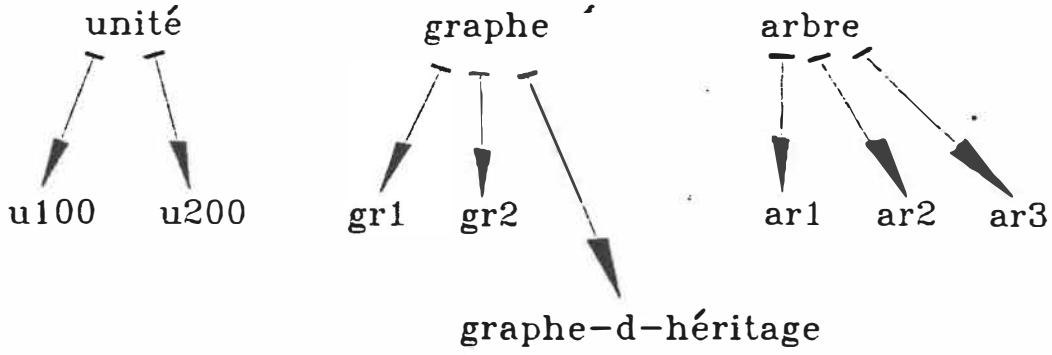

figure I D 1 b : une partie de l'arbre d'instanciation

L'arbre d'instanciation représente le graphe de la relation "a pour instance" entre l'ensemble des classes et celui des unités. Une flèche (avec talon) d'une classe C vers une unité U signifie : "U est une instance de C", à partir de quoi on peut dire : "U est un(e) C". On peut lire ici, par exemple, que `ar1` et `ar2` sont des arbres, et que `gr1`, `gr2` et `graphe-d-héritage` sont des graphes.

En représentant le graphe d'héritage et l'arbre d'instanciation sur un même schéma, on obtient une vision très synthétique de la structure de RLA à un moment donné. Un tel schéma est donné au paragraphe II B 3.

3. les réflexes

Caractéristiques de nombreux langages à objets, les réflexes, appelés aussi "démons", sont disponibles en RLA. Les réflexes sont des méthodes associées à certains attributs d'objets, qui se déclenchent automatiquement suivant les accès en affectation, annulation, ou modification de valeur. Ils permettent, par exemple, de maintenir la cohérence d'un ensemble de données en effectuant plusieurs vérifications pour chaque nouvelle valeur, ou d'effectuer toutes les modifications entraînées en chaîne par la modification d'une seule valeur.

Plusieurs classes d'attributs utilisant des réflexes existent à la base dans RLA : les attributs qui vérifient toutes les valeurs qu'on leur affecte; ceux qui interdisent les modifications (les attributs constants); ceux qui répercutent les modifications entraînées par leur propre modification; ceux qui, si leur valeur est inconnue, déclenchent en cas de besoin une méthode qui permet de la calculer ou de la demander à l'utilisateur; etc...

Les réflexes sont un outil puissant qui permet à l'utilisateur de ne se préoccuper que des tâches les plus importantes, les réflexes faisant le reste automatiquement et sans que cela ne perturbe le déroulement du travail.

B. un méta-langage

Comme nous l'avons écrit en présentant les langages objets, la possibilité qu'a un langage de représenter des langages lui confère une puissance intéressante pour les applications de l'intelligence artificielle. RLA a cette propriété, que l'on peut appeler : "méta-représentation".

1. représenter des langages à objets

a) "bootstrap"

Des notions fondamentales des langages objets sont représentées dans RLA, i.e. il existe plusieurs objets qui symbolisent ces notions.

Ainsi, les types de RLA (par exemple `fix`, `string` ou `objet`) sont des unités. Ce sont des unités de la classe des types : `type`. La notion fondamentale de type est donc bien représentée dans le langage lui-même. Pour créer un nouveau type, il suffit de créer une nouvelle instance de la classe `type`. De même, les classes sont des unités de la classe `classe`. Toutes les instances de `classe` sont des classes du langage.

La classe `type` est une classe d'un genre particulier : les unités auxquelles elle peut donner naissance, étant des types, disposent d'un ensemble de méthodes. La classe de `type` est donc une spécialisation de la classe `classe`. On l'appelle `méta-type`. `méta-type` hérite directement de `classe`. De la même façon, il faut monter d'un niveau pour définir la classe de la classe `classe` : cette classe est une spécialisation de `méta-type`; il s'agit de `méta-classe`. `méta-type` et `méta-classe` sont aussi, bien-sûr,

des objets représentés dans le langage : ce sont des instances de `méta-classe`...

Il apparaît ici, on le sent bien(!), une difficulté fondamentale. Si les éléments du langage capables de créer les objets (par exemple les classes) sont eux-même des objets, comment sont-ils créés? Les relations d'héritage et d'instanciation entre `objet`, `unité`, `type`, `classe`, `méta-type` et `méta-classe` sont au premier abord assez intriquées, et semblent conférer à l'ensemble une structure supportée uniquement par elle-même, une structure qui ne repose sur rien!

C'est vrai. `méta-classe` est créée par elle-même, et hérite de `méta-type`, qui n'existe pas encore, car il est créé par `méta-classe`... Cette création initiale des objets définis en référence circulaire s'appelle le "**bootstrap**". Elle est faite *hors du langage*, à l'aide d'autres moyens. C'est un procédé connu en informatique : le premier assembleur qui a existé a été écrit manuellement avec une suite de bits; le premier compilateur a été écrit avec un assembleur; le bootstrap de RLA est réalisé en Lisp. Hors de ce bootstrap, une fois tous ses objets définis par une opération inaccessible à RLA, RLA existe "comme si" les objets avaient été créés entre eux. Les poules existent comme si elles avaient toujours été créées par les oeufs. Les oeufs aussi... Leurs "bootstraps" ne les gênent en rien dans leurs existences.

On accède à RLA à partir de l'interpréteur de Lisp en chargeant successivement plusieurs fichiers. Ce chargement effectue toutes les définitions du bootstrap. Les fichiers sont chargés dans l'ordre du plus fondamental au plus spécialisé.

b) contrôle de l'héritage

Après le bootstrap, l'héritage défini et utilisé dans RLA est simple (i.e. un type hérite d'un seul autre type). Mais rien n'empêche de définir et d'utiliser un autre type d'héritage. Il est possible, grâce à la propriété de méta-représentation, d'accéder au mécanisme qui régit l'envoi de message, de créer une nouvelle typologie, et de l'utiliser en accord avec une nouvelle sémantique.

Le problème de l'héritage multiple (i.e. un type peut hériter directement de plusieurs types en même temps) est complexe, et demeure aujourd'hui un problème ouvert (Cf. [DUCOURNEAU 89]). Plusieurs sémantiques sont possibles pour la multiplicité de l'héritage; aucune ne semble s'imposer comme étant la plus générale.

Le besoin d'un héritage multiple ne se faisant, pour l'instant, pas sentir pour la réalisation du système expert pour les barrages, seul l'héritage simple a été implanté. La possibilité de le compliquer par la suite demeure.

c) contrôle des méthodes

Les méthodes, éléments indispensables pour l'action des objets, n'échappent pas au "principe des objets" : les méthodes de RLA sont des objets. Plus précisément, ce sont des unités instances de la classe `méthode-directe` ou d'une de ses sous classes. L'envoi de message, responsable de la recherche de la bonne méthode et de son exécution, envoie lui-même le

message exec à la méthode trouvée. (Le cercle vicieux, qui ne manquera pas d'intriguer, est résolu dans le bootstrap.)

Plusieurs classes de méthodes sont définies; elles héritent toutes de la classe de base des méthodes : `méthode-directe`. Chacune a sa façon d'exécuter ses instances (les méthodes), en répondant au message `exec`.

Par exemple, en plus d'exécuter ses méthodes, la classe `méthode-stat` gère des statistiques sur leurs exécutions : statistiques sur les temps d'exécutions et sur le nombre d'erreurs. Ces statistiques, étant des attributs, sont accessibles très facilement.

Un autre exemple de classe de méthodes est la classe `méthode-contrôle`. L'exécution d'une méthode-contrôle s'accompagne de vérifications diverses avant l'exécution (si les paramètres doivent satisfaire certains critères, par exemple), ou après (pour vérifier les effets de l'exécution). D'autre part, un jeu de paramètres et résultats attendus peut être fourni au moment de la création de la méthode. Des bancs d'essais sont alors effectués automatiquement à la définition et à chaque modification de la méthode, et signalent tout écart aux spécifications données. Ce type d'outils se révèle être une aide très puissante, et est à classer dans la "boite à outils" du génie logiciel.

d) contrôle des représentations

Les objets sont des entités relativement indépendantes les unes des autres. De la même façon que le sont les routines des bibliothèques de programmes, seules sont importantes les spécifications *externes* des

objets : leurs buts, les résultats susceptibles d'être produits, les domaines de validité des données, le sens que l'on peut leur donner. La représentation interne d'un objet (ou le code d'une routine) n'est pas utile à son utilisateur; elle n'est connue que de son créateur.

Les choix de représentation des objets, des implantations de routines ou des représentations des données, sont généralement basés sur des critères d'efficacité, que ce soit du point de vue encombrement mémoire ou de la rapidité d'accès ou d'exécution. Dans un cas, telle représentation, même si elle est plus gourmande en place mémoire, sera préférée à telle autre, plus compacte mais plus difficile d'accès. Alors que le choix contraire sera préféré dans un autre cas. Des compromis sont sans cesse imposés. RLA permet non seulement de choisir, créer et modifier les représentations des objets afin de satisfaire les critères fixés, mais aussi de faire coexister plusieurs représentations d'un même objet, afin de profiter des avantages de chacune d'elles. Par exemple, en RLA les ensembles peuvent être représentés par des listes de plusieurs sortes, certaines permettant l'apparition multiple d'éléments, d'autres non, de façon à ce que les méthodes qui agissent sur les ensembles utilisent la représentation la meilleure, celle qui les rend plus efficaces, et puissent passer d'une représentation à une autre selon les besoins. Ces changements de représentations, effectués automatiquement par les méthodes, ou imposés par l'utilisateur, peuvent avoir lieu à n'importe quel moment, même au cours de l'utilisation de la donnée.

Par rapport à l'informatique "classique", RLA apporte donc deux nouveautés supplémentaires : permettre d'une part de choisir et modifier la

représentation des objets, et d'autre part de le faire en cours d'utilisation des objets, ou de faire coexister plusieurs représentations et d'utiliser à un moment donné celle qui offre le plus d'avantages.

2. la propriété d'auto-référence

a) auto-référence et intelligence

L'intelligence est souvent associée, non seulement à la capacité de raisonner, mais aussi à la capacité de raisonner *sur soi-même*. Une des choses qui distingue l'homme des autres êtres vivants est sa conscience de lui-même. Cette capacité est le fondement de la plupart des comportements "intelligents". Par exemple, un homme peut se rendre compte que son raisonnement "tourne en rond", et apporter une solution par une autre voie.

Rappelons ici un des grands résultats fondamentaux démontrés récemment en mathématiques : le théorème d'incomplétude de Gödel. Il affirme en substance que dès qu'un système est assez puissant, il est limité fondamentalement. Un système est "assez puissant" dès qu'il a la capacité d'auto-référence. Il est "limité fondamentalement" dans le sens où des vérités lui échappent, lui demeurent à jamais indémontrables. Plusieurs auteurs ont associé les notions "assez puissant", intelligence et auto-référence. Ce qui rend un système (et en particulier le cerveau humain?) "intelligent" est l'auto-référence; cette auto-référence est aussi la cause de sa limite.

b) auto-référence et intelligence artificielle

Fort de ces constatations, l'intelligence artificielle cherche à construire des systèmes auto-référents, sachant que *jamais* un tel système n'aura réponse à tout, mais que c'est cependant la seule façon de le rendre "intelligent".

Des langages à objets, tels RLA, ont des capacités d'auto-référence. Comme on l'a vu précédemment, les objets de RLA sont regroupés en types qui sont eux-mêmes des objets; les unités sont créées par des classes qui sont des types particuliers; les méthodes sont des unités exécutées par d'autres méthodes qui peuvent exercer des contrôles sur l'exécution.

3. objets et méthodes de base

Les objets de base les plus nombreux disponibles au lancement de RLA sont ceux qui permettent la *création* d'autres objets. Nous avons vu les principaux :

- les classes permettent de créer les unités
- les méta-types permettent de créer les types.
- les méta-classes permettent de créer les classes, les méta-classes et les méta-types

Sur le même principe, les méthodes de base disponibles au début sont les méthodes qui permettent l'apprentissage et la création de nouvelles méthodes. On dispose également de méthodes d'information sur les objets :

?? pour savoir ce qu'est un objet particulier, et menu pour savoir ce que l'on peut lui faire faire.

Le schéma représentant à la fois le graphe d'héritage et l'arbre d'instanciation après le bootstrap de RLA permet de voir tous les objets de base et leurs relations. Ce schéma est donné sur la figure II B 3.

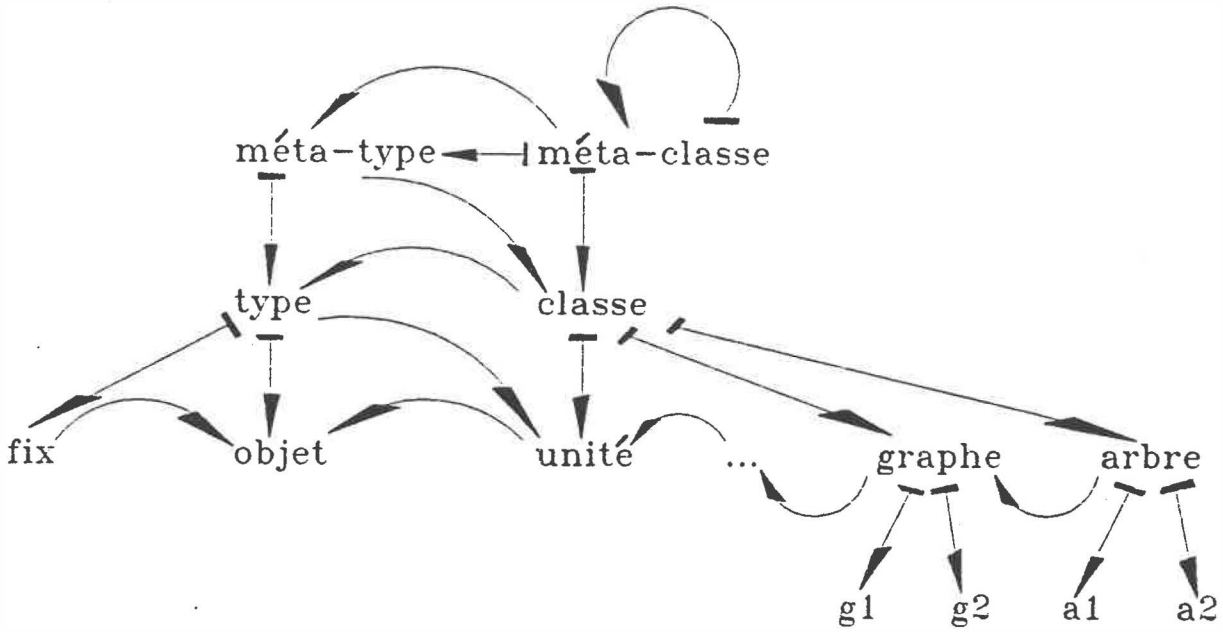

figure II B 3 : graphe d'héritage et arbre d'instanciation de RLA après le bootstrap

C. un langage pour construire des systèmes experts

RLA a donc toutes les caractéristiques d'un langage pour développer des systèmes experts. En plus des propriétés de modularité, expressivité, auto-référence, aide au développement (génie logiciel) et facilité de maintenance conférées par le principe des objets, il laisse entièrement

disponible la puissance d'un langage à part entière pour l'intelligence artificielle : Lisp.

Les paragraphes qui suivent montrent comment sont représentés quelques éléments intervenant dans la construction du système expert.

1. faits

Les faits sont représentés par autant de formes que l'on désire : les caractéristiques de tout objet peuvent être considérées comme autant de faits par le moteur d'inférences.

On représente donc les composantes des barrages par des objets classés dans une hiérarchie de classes spécialement créées : la classe des sols, la classe des barrages, la classe des blocs qui composent le barrage, les classes des filtres, des noyaux, des drains, la classe des instruments de mesure, etc... Ainsi par exemple, le fait "le pourcentage passant le tamis #200 du noyau est 30%" est simplement représenté par la valeur 30 de l'attribut `%-passant-tamis-200` de l'objet noyau, instance de la classe des sols. On le voit dans l'affichage des caractéristiques de l'objet, réponse du message `[noyau ??]` :

```
objet              : noyau
type               : sol
propriétés
#:system:loaded-from-file : barf1.ll
Unité
d-85               : 2
%-passant-le-tamis-4       : 95
%-passant-le-tamis-200     : 30
```

Les objets constituent donc une base de données d'un nouveau genre. Mais il est bien sûr possible d'exploiter des bases de données classiques : les fichiers à accès direct ou séquentiel sont aussi représentés par des objets, et accessibles aussi facilement que les autres objets. Les nombreuses données manipulées par les autres programmes appliqués aux barrages peuvent ainsi être disponibles directement en RLA, sans perte de temps due à une conversion ou une restructuration éventuelle. On pourrait par exemple utiliser les données des observations visuelles gérées par le programme de Smith (Cf. [SMITH 87]).

On a crée pour le système expert, d'autre part, une classe qui sert à représenter les faits qui ne sont pas attachés à un objet, la classe `fait`. Ces faits sont essentiellement constitués de l'énoncé du fait. Voici un exemple d'une instance de la classe `fait` :

```
objet              : fait2
type               : fait
propriétés
#:system:loaded-from-file : gc
Unité
énoncé             : (critère ok pour le sol s1 et le filtre
                     f2)
certitude          : (0.5 1)      ; (Cf. explications plus loin)
source             : la
date               : sept 89
```

2. règles

Une des formes de connaissances utilisées par les experts est la règle. Dans la mesure où une règle peut être interprétée logiquement, on peut la représenter avec deux parties, prémisses et conclusions, et l'utiliser en

accord avec les principes de déduction de la logique classique (ou d'autres logiques étendues, pour prendre en compte l'imprécision et l'incertitude, par exemple). Les règles sont des connaissances de base de nombreux domaines intervenant dans le système expert, aussi bien en géotechnique qu'en mathématiques et en physique.

De telles règles sont par exemple :

- si un sol subit des tassements différentiels
  alors le sol qui est au dessus a de grandes chances de se fissurer
- si un sol est du groupe 3, et
  le d85 du sol est D85, et
  le grand-d15 du filtre est D15, et
  $D15 \leq 4 \times D85$
  alors le critère de Sherard est vérifié pour ce sol et ce filtre
- si un sol et un filtre ne vérifient pas le critère de Sherard
  alors il est probable que le filtre ne remplisse pas correctement un de ses rôles

etc...

La deuxième règle, par exemple, est représentée par l'objet :

```
objet            : règle3
type             : règle
propriétés
#:system:loaded-from-file : barr1.ll
Unité
définition       : critère pour sol de groupe 3
prémisses        : ((Sol est du groupe 3)
                    (le grand-d-15 de Filtre est Grand-d-15)
                    (le d-85 de Sol est D-85)
                    (vérifie (<= Grand-d-15 (* 4 D-85))))
conclusions      : (((critère ok pour le sol Sol et le
                      filtre Filtre)
                     (1 1)))
source           : sherard
date             : 88
variables        : (Sol Filtre Grand-d-15 D-85)
```

On peut dès maintenant remarquer que ces règles ont des variables. La deuxième, par exemple, peut s'appliquer à n'importe quel sol : les mots "un sol" (représentés dans la forme RLA de la règle par une variable : un symbole qui commence par une majuscule), représentent tous les sols potentiels. On distinguera donc les symboles `d-85` et `D-85` : `d-85` est une "constante" qui définit le texte de la prémisse, alors que `D-85` est une variable qui sera instanciée par des valeurs (numériques ici). Ces valeurs seront utilisées pour effectuer les vérifications de la relation numérique de la prémisse `(vérifie (<= Grand-d-15 (* 4 D-85)))`. Remarquons au passage la notation préfixée utilisée en Lisp, en RLA, et donc aussi dans nos règles : `(<= Grand-d-15 (* 4 D-85))` signifie en écriture mathématique habituelle : `Grand-d-15 ≤ (4 * D-85)`.

Ce type de règle est plus difficile à traiter, mais aussi beaucoup plus avantageux, que les règles sans variables, comme on le verra plus loin au chapitre III.

3. méta-règles

Les méta-règles sont des règles qui portent sur le domaine de l'utilisation des règles. Mais ce sont avant tout de simples règles. Elles sont représentées exactement de la même façon que les règles "classiques", avec prémisses, conclusions, variables, comme les instances de la classe `règle`.

S'il est évident que l'expert utilise des règles de son domaine d'expertise, il est moins évident --mais tout aussi vrai-- qu'il utilise aussi des règles qui guident son raisonnement et qui le rendent efficace. Certaines de ces méta-règles sont spécifiques aux règles du domaine étudié initialement (ici les barrages), d'autres concernent l'utilisation des règles *en général*. Ces dernières sont l'objet d'étude des cogniticiens, experts dans le domaine de la connaissance.

Nous verrons plus précisément au chapitre III à quoi peuvent servir les méta-règles. En voici déjà quelques exemples :

- si une règle n'est jamais utilisée
  alors il est probable qu'on puisse l'oublier
- si une règle est très souvent utilisée, et
  on cherche à résoudre un problème,
  alors tester avant les autres si elle s'applique à ce problème

- si "on boucle" au cours d'un raisonnement
  alors interrompre le raisonnement

4. bases de connaissances

Qu'est-ce qu'une base de connaissances? Réponse : un ensemble de connaissances. La représentation des bases de connaissances s'impose donc : on utilisera le type ensemble de RLA, après une spécialisation éventuelle qui permettra d'attacher des méthodes spécifiques pour les connaissances. On peut ainsi utiliser des bases de faits, des bases de règles, des bases de méta-règles, des bases mixtes...

Voici, par exemple, les bases de faits et de règles du moteur m1 :

```
[m1 ? base-de-faits] = {fait2}

[m1 ? base-de-règles] = {règle12 règle11 règle10 règle9
règle8 règle7 règle6 règle5 règle4 règle3 règle2 règle1}
```

5. moteurs

Les moteurs d'inférences n'échappent pas à la règle désormais incontournable : ce sont des objets. Voici le moteur m1 :

```
objet              : m1
type               : moteur
propriétés
#:system:loaded-from-file : barmot1.ll
Unité
définition         : un premier moteur pour les barrages
base-de-faits      : {fait2}
base-de-règles     : {règle12 règle11 règle10 règle9 règle8
                      règle7 règle6 règle5 règle4 règle3
                      règle2 règle1}
```

Instances d'une classe de moteurs, plusieurs moteurs peuvent exister en même temps, chacun ayant ses propres bases de connaissances (éventuellement partagées). Plusieurs classes de moteurs peuvent aussi exister simultanément; on dispose alors de plusieurs façons de concevoir les moteurs, de plusieurs mécanismes possibles. Moteurs d'ordre zéro, d'ordre 1, avec ou sans traitement de l'incertitude, en chaînage avant, arrière ou mixte, tous peuvent exister simultanément et être utilisés concurremment.

Les applications possibles de collaboration entre plusieurs moteurs restent à concevoir... Pour notre système expert, un seul type de moteur à été réalisé, mais qui tente d'incorporer de nombreux atouts : ordre 1 (i.e. présence de variables dans les règles), chaînage arrière, traitement de l'incertitude, utilisation possible des méta-connaissances. Nous le détaillerons au chapitre III.

D. objectifs atteints

Le langage de représentation et manipulation de connaissances, indispensable outil de construction du système expert, est maintenant réalisé. Ses caractéristiques répondent à toutes les exigences imposées par un tel projet.

1. expressivité

Tous les types de connaissances énumérés au chapitre I B sont représentables en RLA. D'autre part, RLA est un langage qui dispose de

toute la puissance d'un langage informatique complet : Lisp. C'est donc un langage très expressif, capable de traiter une multitude de domaines, hydraulique, géotechnique, barrages, connaissances, langages de représentation...

2. introspection et explication du comportement

La propriété d'auto-référence de RLA lui permet en particulier de rendre compte de son comportement, et confère au système expert la possibilité d'expliquer son raisonnement à tout moment. Ce point est essentiel. La confiance aveugle qu'a généralement l'utilisateur d'une calculatrice ou d'un programme de calcul numérique disparaît quand le programme manipule des notions symboliques d'un plus haut niveau d'abstraction, tels les systèmes experts. Tout fait inféré doit pouvoir être justifié sur demande de l'utilisateur.

3. utilisation interactive

On utilise Lisp et RLA au moyen d'un interpréteur, ou "boucle d'interaction" : indéfiniment, les demandes de l'utilisateur sont lues au clavier, puis exécutées, et les résultats produits sont affichés à l'écran. Ceci n'empêche en rien la compilation de la plupart des programmes et méthodes; le compilateur de Lisp est dynamique, automatique ou sur demande. Lisp (et donc RLA) est, contrairement à l'idée souvent reçue qui prétend que ce n'est pas possible, un interpréteur *et* un compilateur.

RLA étant un langage objets, l'action de base est l'envoi de message. C'est donc aussi la forme de communication que doit adopter l'utilisateur.

Signalons ici deux méthodes d'information universelles (i.e. qui s'appliquent à *tout* objet) : ?? et menu. Le message [*objet* ??] affiche des informations sur la nature et l'état de l'objet *objet*; le message [*objet* menu] renvoie la liste de tous les messages que l'on peut envoyer à *objet*.

4. modularité

La "philosophie des objets" confère au système qui l'utilise une grande modularité. Les différents éléments sont créés un à un, dans un ordre qui va du plus simple au plus complexe, du plus général au plus spécialisé, mais laissant cependant une certaine autonomie à chacun. Chaque objet peut être considéré seul, et observé ou modifié relativement indépendamment des autres. Un objet peut être considéré par les autres comme une "boite noire", définie uniquement par ses spécifications, et dont la machinerie interne, i.e. la représentation physique et le détail de l'implantation des méthodes, est inaccessible. C'est le principe de **transparence**.

Ces principes de transparence et de modularité, que l'on retrouve dans les autres grands langages (tels Pascal, C, et surtout Ada), permet de réduire très sensiblement la complexité du développement et de la maintenance requises habituellement par les logiciels de grande taille.

RLA s'avère efficace en particulier pour le développement de maquettes, systèmes exigeant sans cesse des modifications, ajouts et suppressions, selon la progression des recherches et des résultats obtenus. Le système expert sur les barrages développé dans le cadre du présent projet peut être considéré comme une maquette, démontrant *rapidement* la faisabilité d'un tel système à une autre échelle.

III. moteur d'inférences

Nous disposons maintenant d'un langage répondant aux deux objectifs :

1. pouvoir représenter toutes les connaissances nécessaires au système expert,
2. être un langage de programmation des algorithmes du système expert.

Le présent chapitre est consacré à la définition et la réalisation d'un moteur d'inférences. Que permet de faire l'informatique? Qu'apporte l'intelligence artificielle? Quelles caractéristiques choisir pour notre moteur? Réalisation et utilisation dans le domaine des barrages.

A. résolution de problèmes par la machine : état de l'art

Dans une perspective de justification des moyens développés pour notre système expert, nous présentons ici rapidement les différentes possibilités que nous offre l'informatique, leurs apports et leurs limites.

1. informatique classique

Jeune de quelques dizaines d'années, l'informatique s'est surtout développée autour des méthodes de calcul numérique. Les outils informatiques, tant matériels que logiciels, qui existent aujourd'hui mettent à notre disposition des puissances considérables de calcul.

Ne citons qu'un exemple parmi la variété quasi infinie des utilisations : la conception assistée par ordinateur (C.A.O.) pour les projets de génie civil développée dans le cadre du projet Castor à l'Ecole Polytechnique de Montréal (Cf. par exemple [CAMARERO 88]). Le fonctionnement de ces programmes, non seulement les simulations de comportement, mais aussi ce qui concerne la saisie et l'affichage graphiques des données et résultats, est basé sur une représentation numérique et une simulation du "monde" considéré : les objets à traiter (barrages, sols, turbines et autres structures) sont découpés en un grand nombre d'éléments (cette méthode est appelée "méthode des éléments finis"), dont on connait ou on estime les caractéristiques moyennes. A partir de ce découpage sont formées de grands systèmes linéaires ou non (plus de 1000 inconnues) traités par des algorithmes itératifs de calcul. Plus on passe de temps à calculer, plus "on fait de boucles", et plus les résultats de la simulation sont précis et valables à long terme.

Ce projet réalise l'intégration de plusieurs modules pluridisciplinaires dans un cadre commun de développement (Cf. [CAMARERO 88]) lui conférant une homogénéité, une qualité et une efficacité uniques. La principale limitation cependant, de cette application de pointe de l'informatique "classique", est fondamentale. Les algorithmes utilisés ne tiennent pas compte des particularités des données; ils s'exécutent de la même façon *quelles que soient* les données. Par exemple, un programme de calcul de produit de matrices effectuera ses multiples boucles même si une des matrices facteur est nulle, auquel cas le résultat pourrait s'obtenir bien plus rapidement que par le calcul : par raisonnement. On mesure cette

"gourmandise" de ressources des algorithmes par ce que l'on appelle la **complexité**. Le temps et l'espace mémoire nécessaires croissent avec le carré ou même l'exponentielle de la taille des données initiales et de la précision désirée. Une complexité exponentielle signifie que les améliorations des performances des machines et programmes se font sentir beaucoup moins vite au niveau de la précision et de la rapidité des résultats.

2. intelligence artificielle

L'intelligence artificielle tente de repousser les limites de ce que les ordinateurs peuvent résoudre en changeant l'approche des problèmes. D'abord, elle cherche à utiliser des **symboles** autres que des nombres pour représenter les éléments du problème. Cette représentation est très souvent plus proche de notre façon courante de nous représenter les choses. On se représente plus volontiers et plus naturellement un barrage comme un assemblage de fondation, noyau, filtres, recharges, drains avec chacun leurs caractéristiques, que par une matrice numérique 500 x 500.

Ensuite, les algorithmes utilisés en intelligence artificielle tentent de profiter des particularités des données pour gagner du temps quand cela est possible. Par exemple, avant d'effectuer un produit complexe, un tel algorithme pourra tester la nullité d'un des termes, évitant ainsi une perte de temps éventuelle. Remarquons que cette démarche est proche de la façon de faire humaine.

Enfin, pour donner de telles capacités aux programmes, l'intelligence artificielle cherche à reproduire dans une certaine mesure les

connaissances et le savoir-faire de l'homme, en essayant de les expliciter morceau par morceau. Ainsi, petit à petit, les connaissances informatisées s'accumulent en bases de connaissances dont le rôle est bien plus fondamental que celui des algorithmes qui les utilisent.

L'intelligence artificielle, au début très ambitieuse, s'attaquait à des problèmes généraux, tels la traduction de textes d'une langue dans une autre, ou la réalisation d'un "programme capable de résoudre des problèmes généraux" (general problem solver, ou GPS). Mais les difficultés rencontrées ont rapidement conduit à écrire des systèmes appliqués à des domaines bien définis et délimités. La taille des bases de connaissances est ainsi devenue à peu près raisonnable. Ces systèmes traitant des domaines spécialisés et construits autour de programmes et de bases de données s'appellent des systèmes experts; ils sont aujourd'hui l'application la plus répandue de l'intelligence artificielle.

3. moteurs d'inférences

Le principe des systèmes experts est simple; ils sont composés de deux parties :

- une ou plusieurs **bases de connaissances** (données, faits, règles, méta-règles), éléments fondamentaux pour le succès du système expert, et aussi le plus important en taille,
- un **"moteur d'inférences"** : relativement petit programme capable d'enrichir la base de faits avec de nouveaux faits créés à partir de ceux déjà connus.

Il existe plusieurs façons de créer des nouveaux faits, donnant naissance à au moins autant de types de moteurs d'inférences différents.

a) logique des propositions

Le moteur le plus simple est basé sur la théorie mathématique de la logique propositionnelle. Les connaissances sont représentées par des propositions : suites de symboles. Par exemple : "le %-passant-le-tamis-200 de filtre-aval est 30", ou "le critère de filtre de Sherard est OK pour noyau et filtre-aval". Pour inférer des faits, on utilise des règles composées de prémisses et de conclusions. Les prémisses et conclusions sont aussi des propositions. Voici un exemple de règle en logique propositionnelle :

règle1 :

si	noyau1 est de type 3
et	le %-passant-le-tamis-200 de filtre-aval est inférieur à 30
et	...
alors	le critère de filtre de Sherard est OK pour noyau1 et filtre-aval

Le mécanisme de déduction utilisé est extrêmement simple; c'est le Modus Ponens : si on a "A" et "si A, alors B", alors on peut déduire "B". Il s'écrit aussi de la façon suivante :

Modus Ponens (MP) :

si A

et A -> B

---

alors B

Les systèmes de ce type, appelés moteurs d'ordre 0 (ou sans variables), ne sont pas d'une grande utilité pratique car ils nécessitent un nombre bien trop important de règles comparé à la taille des résultats. Ainsi, la règle1 de notre exemple ne pourra jamais créer qu'un seul fait : "le critère de filtre de Sherard est OK pour noyau1 et filtre-aval". Pour tester le critère de Sherard sur un autre filtre par exemple, il faut disposer d'une autre règle avec le nom de l'autre filtre; pour le tester avec un autre noyau, il faut encore une autre règle...

On voit donc bien la nécessité d'utiliser des variables : en remplaçant les symboles "noyau1" et "filtre-aval" par des variables pouvant représenter n'importe quels noyaux et filtres, on pourrait utiliser la règle1 pour vérifier le critère de Sherard sur tous les types de noyaux et de filtres.

b) logique des prédicats

L'extension de la logique de propositions à l'utilisation des variables s'appelle la logique des prédicats, ou logique d'ordre 1. Beaucoup de variantes de cette logique sont utilisées en intelligence artificielle.

Prolog est le langage informatique dont le fonctionnement, même si il est limité à un sous-ensemble de cette logique d'ordre 1, se rapproche le plus rigoureusement de la théorie. (On parle ici de Prolog *pur*, sans les artifices procéduraux qui accompagnent généralement son utilisation et qui, croyant bien faire, entraînent souvent un des rares langages déclaratifs du marché dans un compromis qui lui ôte son plus grand intérêt.) Ceci lui confère une puissance et une robustesse très importantes pour résoudre certains problèmes, mais c'est aussi la cause de la limitation du nombre d'applications qu'il est susceptible de résoudre : le carcan est souvent trop rigide. La plupart des problèmes et méthodes de raisonnement humains se formalisent mal, voire pas du tout, avec simplement la logique des prédicats.

c) systèmes de production

Plus généraux que les systèmes utilisant strictement la logique d'ordre 1, les systèmes de production (au sens large) sont aussi basés sur l'utilisation de règles avec variables. Ils permettent cependant d'utiliser d'autres systèmes de représentation que les seules propositions ou règles avec variables, et d'autres systèmes de production que le Modus Ponens. Grâce à eux, on peut envisager de faire des inductions, des généralisations, d'apprendre à partir d'exemples ou d'erreurs, de gérer un certain "flou"

dans les données (incertitude et imprécision), d'utiliser des connaissances pour guider le raisonnement, ou pour gérer les connaissances, etc...

La plupart des systèmes experts sont de ce type, chacun combinant un plus ou moins grand nombre de ces caractéristiques.

4. incertitude et imprécision

Avec l'amélioration des systèmes experts, l'incertitude et l'imprécision se révèlent des caractéristiques pratiquement indispensables à représenter et à utiliser. Les connaissances, autant les faits simples que les règles, sont en majorité imprécises ou incertaines, et il faut en tenir compte dans nos systèmes informatiques, sans quoi les résultats seraient trop éloignés de la réalité, ou sans intérêt car trop catégoriques.

L'imprécision est la caractéristique des connaissances vagues, décrites à l'aide de qualificatifs tels que : à peu près, bien plus que, autour de, environ etc... De telles valeurs sont partiellement ou grossièrement spécifiées.

L'incertitude concerne la vérité. Une connaissance est incertaine si l'on n'est pas totalement sûr qu'elle est vraie ou qu'elle est fausse.

Il y a des connaissances précises et certaines ("2 x 2 = 4"), imprécises et certaines ("le pourcentage passant le tamis 200 de tel sol est 30 *à plus ou moins 10%*"), imprécises et incertaines ("la granulométrie de tel sol est *probablement étalée*"), ou précises et incertaines ("il est *probable* que le tassement différentiel maximum du noyau n'a pas atteint 20 cm").

Plusieurs représentations de l'incertitude ont été expérimentées et sont utilisées dans différents systèmes experts. Les moyens les plus connu sont l'utilisation de coefficients de vraisemblance, ou la théorie de l'"endorsement". Il en existe d'autres. Celle que nous avons choisi d'utiliser pour notre système expert est basée sur la théorie des possibilités (Cf. [DUBOIS 85]). Elle est décrite plus loin.

Peu de systèmes, en revanche, représentent des connaissances imprécises. Citons les travaux de H. Prade et R. Martin-Clouaire [MARTIN 85a], qui appliquent la théorie des ensembles flous et représentent et traitent des connaissances imprécises à l'aide d'un moteur général.

5. la méta-connaissance

La méta-connaissance, connaissance sur la connaissance, est utilisée de diverses façons et dans différents buts aujourd'hui en intelligence artificielle.

Par exemple, R. Martin-Clouaire [MARTIN 85b] l'utilise dans son système expert conçu pour aider les géologues dans la résolution des problèmes de migration d'hydrocarbures. La méta-connaissance sert ici deux buts : d'une part, pouvoir aborder le problème à un niveau assez haut d'abstraction, et d'autre part, permettre la prise en compte d'un savoir-faire procédural.

Une autre approche est celle de R. Davis (Cf. ses articles fondamentaux [DAVIS 80a] et [DAVIS 80b]). Davis étudie de façon

générale et approfondie l'utilisation possible de la méta-connaissance pour permettre le raisonnement sur le contrôle de l'utilisation des connaissances, et en particulier ce qui concerne le raisonnement sur les règles de production. Son but est de *rendre efficace* le processus de sélection des connaissances à utiliser pour la résolution d'un problème, résolvant ainsi le problème de l'explosion combinatoire des algorithmes traditionnels. Pour cela, il utilise des méta-connaissances capables d'accéder au contenu des règles, de les interpréter et de les classer suivant des critères définis dynamiquement au cours de la résolution. Les principes qu'il préconise et utilise sont importants et assez généraux pour nous être d'une grande utilité. Parmi tous ceux qu'il donne, relevons seulement l'uniformité de représentation, l'uniformité du mécanisme d'inférences, l'accessibilité totale des connaissances au système, la référence des règles par leur contenu. Ces principes apportent aux systèmes experts flexibilité, fiabilité, souplesse de croissance etc...

B. réalisation

Certaines caractéristiques se révèlent pratiquement indispensables au système expert que nous voulons développer : traiter des informations symboliques, être le plus déclaratif possible, posséder un bon système de déduction (ou d'inférence) avec variables et en chaînage arrière, gérer l'incertitude, et utiliser la méta-connaissance de façon à être le plus "expert" possible. Le système développé tente de les incorporer toutes, ou, tout au moins de permettre à court terme de les incorporer (en particulier

l'utilisation efficace des méta-connaissances, qui n'a pu être développée complètement faute de temps).

1. choix pour notre système expert

Il n'est plus la peine ici de justifier l'approche de notre problème par l'intelligence artificielle plutôt que par l'informatique traditionnelle; ceci est fait depuis le début de ce mémoire. Justifions maintenant cependant les choix subséquents.

Le moyen de représentation des connaissances a été choisi le plus général possible, et un outil a été crée pour répondre à ses exigences : RLA.

Des choix s'imposent quant à la réalisation du moteur d'inférences. Une représentation qui permet l'utilisation des variables dans les règles de production et éventuellement les autres types de connaissances s'impose par sa puissance. Si les algorithmes de traitement sont plus délicats à réaliser, les avantages apportés sont indispensables : généralité, "déclarativité", indépendance relative, facilité de maintenance et de développement des connaissances.

D'autre part, nous avons jugé utile --et même indispensable à moyen et à long termes-- de permettre au système de traiter l'incertitude. Nous développons dans la partie suivante notre façon de faire (empruntée à R. Martin-Clouaire et H. Prade).

Enfin, même si la première version du système expert réalisée fonctionne bien sans méta-connaissance, il a été observé que son emploi s'avérerait indispensable dans la suite proche à donner à son

développement. Certains comportements du moteur, par exemple les bouclages, recherches inutiles parce que déjà effectuées ou sans solution, gestion arbitraire des faits produits, et autres pertes de temps diverses, sont améliorables grâce à l'utilisation de méta-connaissances.

Tous les moyens existent déjà pour représenter cette méta-connaissance; ils ont été prévus dès le début du projet. La principale difficulté réside dans leur énoncé. Quelles règles (ou autre forme de représentation?) énoncer et comment les utiliser pour éviter le bouclage, par exemple, ou pour gérer convenablement les bases de faits? Ces questions sont aujourd'hui, avec beaucoup d'autres, du domaine de la recherche. Ce domaine constitue un domaine d'expertise à part entière; on peut l'appeler la "cognitique". On peut se référer à [DAVIS 80a] et à [DAVIS 80b] pour voir une approche très intéressante de l'utilisation des méta-connaissances appliquées au contrôle des programmes.

2. traitement de l'incertitude

Nous avons choisi de représenter l'incertitude dans notre système grâce à la théorie des possibilités. Des fonctionnalités adaptées seront inclues dans notre moteur.

Nous présentons ici sommairement la théorie des possibilités; on peut se référer à [DUBOIS 85] pour une description plus complète. De même, nous présentons la façon dont notre moteur propage l'incertitude, inspirée fortement du moteur SPII-1 décrit dans [MARTIN 85a].

a) théorie des possibilités

La théorie des possibilités naît de plusieurs constations, en particulier :

- Quand la complexité d'un système croît, il apparaît un seuil à partir duquel précision et sens s'excluent mutuellement. En d'autres termes, la précision nuit au sens que l'on peut trouver dans les résultats.
- La prise en compte de l'incertitude et de l'imprécision ne nuit pas à la rigueur. Il suffit de formaliser correctement ces notions et de définir une sémantique sans ambiguïté.

La théorie des possibilités fournit des moyens de représenter l'imprécision (avec les ensembles flous) et l'incertitude (avec les mesures de confiance que sont la nécessité et la possibilité). Les ensembles flous, non utilisés encore dans notre système, ne sont pas abordés ici.

A un ensemble d'événements, on peut associer des mesures de confiance M, i.e. des fonctions qui vérifient :

- les valeurs prises par M sont dans l'intervalle réel [0, 1]
- si l'événement E est certain, alors $M(E) = 1$
- si l'événement E est impossible, alors $M(E) = 0$
- M est croissante, i.e. si $E => F$, alors $M(E) \leq M(F)$
  (qui signifie : "si E implique F, alors on a au moins autant confiance en F qu'en E")

Les notions intuitives de possibilité et de nécessité peuvent se représenter par des mesures de confiance. La mesure de **possibilité $\pi(E)$** d'un événement E est un nombre qui représente à quel point on est sûr que E est vrai. La mesure de **nécessité N(E)** d'un événement E est un nombre qui représente à quel point on est sûr de l'impossibilité du contraire de E. Pour un même événement E, $\pi(E)$ et N(E) sont très peu liées; elles vérifient les relations "faibles" suivantes (où E et F sont deux événements quelconques) :

- $N(E) \leq \pi(E)$ (un événement est au moins autant possible que nécessaire)
- $N(E) > 0 \Rightarrow \pi(E) = 1$ (pour qu'un fait soit nécessaire, il faut qu'il soit complètement possible)
- $\pi(E) < 1 \Rightarrow N(E) = 0$ (contraposée de la relation précédente)
- $N(E) = 1 - \pi(\text{non } E)$ (relation entre possibilité et nécessité)
- $\pi(E \text{ ou } F) = \max(\pi(E), \pi(F))$
- $N(E \text{ et } F) = \min(N(E), N(F))$

L'incertitude d'un événement est donc représentée par un couple $(N(E)\ \pi(E))$, dont les valeurs s'échelonnent ainsi :

(0 0) E est absolument faux

...

(0 0.5) E est quelque peu possible

...

(0 1) ni E, ni son contraire ne sont nécessaires. C'est la plus grande incertitude que l'on puisse avoir sur un événement.

...

(0.1 1) E est assez probable

...

(0.5 1) E est pratiquement certain

...

(1 1) E est certain.

Les correspondances entre les couples de valeurs et les expressions du sens commun sont arbitraires et n'ont pas besoin d'être très précises (sauf pour les couples (0 0) et (1 1)).

b) application au système expert

(1) faits

Tout fait peut être accompagné d'un couple (N π) qui caractérise sa certitude. Un fait qui n'a pas de tel couple est considéré comme certain.

(2) règles

Une règle "si A, alors B" (notée A -> B) est incertaine dans la mesure où l'on n'est pas complètement sûr que, si A est vrai, alors B l'est aussi. On

peut mesurer cette incertitude en généralisant les notions de condition nécessaire et condition suffisante.

N(A -> B) évalue à quel point on est certain de pouvoir déduire B à partir de A., ou encore, à quel point il est nécessaire que B soit vrai pour que A soit vrai (sic). Ainsi, par exemple, si N(A -> B) = 1., alors il est complètement nécessaire que B soit vrai dès que A l'est; alors que si N(A -> B) = 0, il n'est pas du tout nécessaire que B soit vrai si A l'est, et on peut avoir A et non B.

π(A -> B) est définie par une des relations énoncées précédemment :

$$\pi(A \rightarrow B) = 1 - N(\text{non}(A \rightarrow B))$$
$$= 1 - N(\text{non } A \rightarrow \text{non } B)$$

Dans les bases de notre système expert, à chaque règle est associé un couple (N π).

c) propagation de l'incertitude par le moteur

Un schéma de raisonnement est établi à partir des propriétés axiomatiques de la théorie des possibilités. Il s'agit d'une version

généralisée à l'emploi de l'incertitude du Modus Ponens. Voici comment il s'énonce :

Modus Ponens généralisé (MPG) :

si $N(A) \geq a$ et $\pi(A) \leq a'$

et $N(A \rightarrow B) \geq b$ et $\pi(A \rightarrow B) \leq b'$

---

alors $N(B) \geq \min(a, b)$ et $\pi(B) \leq \max(a', b')$

ou encore (version simplifiée) :

si $N(A) = a$ et $\pi(A) = a'$

et $N(A \rightarrow B) = b$ et $\pi(A \rightarrow B) = b'$

---

alors $N(B) = \min(a, b)$ et $\pi(B) = \max(a', b')$

Si une règle comporte plusieurs prémisses A_i, leurs couples $(N(A_i)\ \pi(A_i))$ associés sont combinés pour former un couple unique $(N(A)\ \pi(A))$ avant d'être utilisé par le Modus Ponens généralisé :

$$N(A) = \min N(A_i)$$

et $$\pi(A) = \max \pi(A_i)$$

Si une règle a plusieurs conclusions B_i, à chacune d'entre-elles est associé un couple $(N(A \rightarrow B_i)\ \ \pi(A \rightarrow B_i))$.

d) combinaison des incertitudes d'un même fait

Il arrive qu'un même fait A soit déduit de plusieurs façons différentes ou apparaisse plusieurs fois dans les bases de faits, avec des certitudes qui n'ont aucune raison d'être les mêmes. Ceci semble ne pas avoir de sens. Quelle est *la* certitude de ce fait? Cette certitude, le couple $(N(A)\ \pi(A))$, est obtenue par combinaison des différentes certitudes existantes $(N(A)_i\ \pi(A)_i)$ grâce à des formules données dans [MARTIN 85a] (formules (16) p. 361).

A chaque fois que le moteur produit un nouveau fait, il vérifie que celui-ci n'existe pas déjà. Si il existe déjà, le moteur combine les différentes certitudes et ne conserve qu'une seule "version" du fait.

3. moteur d'ordre 1 en chaînage arrière

Nous avons réalisé, dans le langage RLA, un moteur d'ordre 1 en chaînage arrière. Il fonctionne avec des bases de faits et de règles quelconques. Par principe, un moteur d'ordre 1 en chaînage arrière s'utilise par questionnement : on lui pose des questions auxquelles il répond sous forme de faits inférés.

"**Ordre 1**" signifie que les questions que l'on peut poser à ce moteur sont des énoncés de faits qui peuvent comporter des variables. Si une question ne comporte pas de variable, le moteur essaye de démontrer le fait à l'aide de toutes les connaissances dont il dispose dans ses bases. Il répond en créant le fait et en annonçant le degré de certitude qu'il lui accorde : de (0 0) pour certainement faux, à (1 1) pour certainement vrai. L'utilisateur

peut alors aussi consulter à sa guise toutes les étapes du cheminement du moteur : quelles règles ont été utilisées? quels faits? quels raisonnements? Le moteur est ainsi en mesure de *justifier tout ce qu'il produit.*

Si aucune règle ou fait ne peut apporter la moindre solution à la question, le moteur ne peut pas produire de fait, et il s'arrête et ne répond rien.

Si une question comporte des variables, le moteur établit toutes les combinaisons de valeurs possibles à donner aux variables, et pour chacune d'elles, répond à la question instanciée (i.e. ne comportant plus de variable) de la même façon que précédemment. La réponse du moteur à une question avec variables est donc constituée d'une liste d'associations variable/valeur, avec pour chacune d'elles, une certitude et un fait inféré.

"**En chaînage arrière**" signifie que pour répondre à une question de l'utilisateur, le moteur peut être amené à répondre à d'autres questions. L'algorithme du moteur d'inférences est donc un algorithme récursif : il s'appelle lui-même. Ainsi par exemple, si la question de l'utilisateur est "Est-ce que tel sol et tel filtre vérifient le critère de Sherard?", le moteur sera amené à se poser la question "De quel type est tel sol?". Après avoir déterminé le type du sol en question (opération qui peut elle-aussi nécessiter d'autres questionnements récursifs), le moteur pourra répondre à la question initiale. En général, les questions que le moteur se pose récursivement possèdent des variables, même si la question initiale de l'utilisateur n'en a pas.

C. exemple concret : les critères de filtre de Sherard

1. intérêt

Afin de bien montrer comment ont été appliquées toutes les recherches et de donner un aperçu de ce dont est capable le système expert dans sa version actuelle, voici un exemple complet d'utilisation. Cet exemple est volontairement assez simple dans le but d'être compréhensible rapidement. Il est cependant plus utile qu'un simple cas d'école et peut être utilisé tel quel dans des applications de plus large envergure.

Le problème considéré est la vérification des critères de filtre de Sherard, tels qu'ils sont définis dans son article sur les filtres et le contrôle des écoulements dans les barrages en terre [SHERARD 85]. Les études de Sherard font référence dans le domaine des barrages.

Sherard, dans cet article, critique le concept de "défense multi-lignes" ("multi lines of defense") contre les écoulements et les fuites dans les barrages en terre. Cette méthode généralement utilisée dans la conception des ouvrages de retenue est basée sur le fait que l'on ne connait aucune ligne de défense dans laquelle on a complètement confiance, et que l'on croit qu'en en utilisant plusieurs différentes de front, le barrage sera plus protégé. Sherard s'élève contre cette pratique qui, selon lui, est due à un manque de connaissance, d'où un manque de confiance.

Il a donc étudié de façon approfondie ce qu'il croit être une méthode *sûre* : les filtres. Ses études ont abouti à l'énoncé de critères qui, si ils sont satisfaits, garantissent un bon comportement des barrages.

L'expert Sherard a énoncé des critères que l'on va représenter sous forme de règles et utiliser dans notre système expert. Les critères en question, au nombre de quatre (un par type de sol), sont donnés dans [SHERARD 85] aux pages 9 à 11 et résumés à la page 27.

2. contexte de l'exemple

Situons notre exemple d'utilisation du système expert pour des critères de filtre à l'intérieur d'un problème plus grand. Les démarches des experts décrites ici seront, à terme, effectuées par le système lui-même.

Considérons le cas du barrage dont une coupe schématique est donnée sur la figure III C 2. Plusieurs instruments, dont un piézomètre installé dans le noyau, auscultent le barrage en permanence. Au cours de l'exploitation du réservoir, on se rend compte que les mesures de charge d'eau fournies par ce piézomètre deviennent trop élevées par rapport aux valeurs attendues. Les experts sont donc amenés à rechercher les causes de ce malfonctionnement.

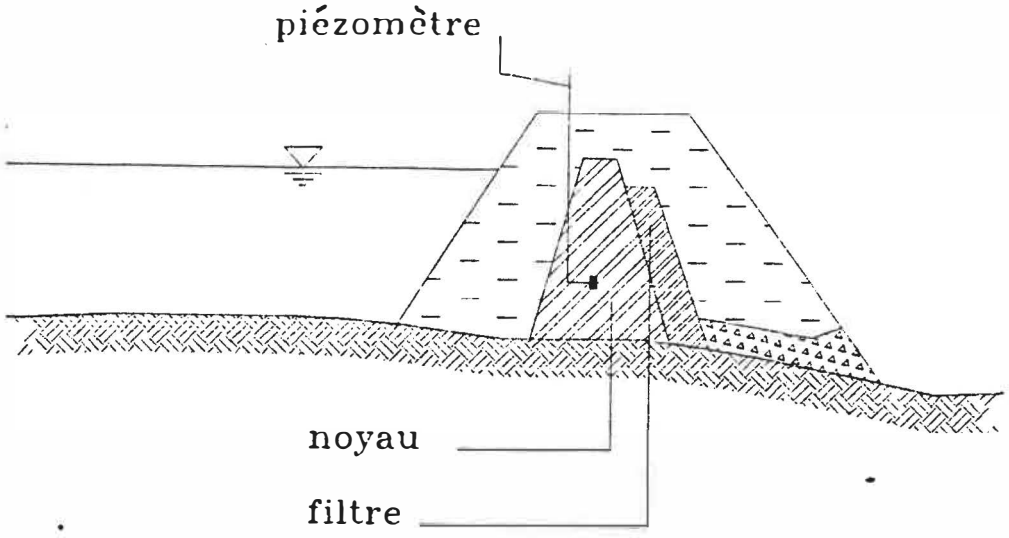

figure III C 2 : coupe schématique du barrage pris comme exemple d'étude

Les experts émettent de suite des hypothèses quant à l'origine du problème. Ils utilisent leur connaissance des rôles fondamentaux des objets impliqués :

le noyau :

- retenir l'eau

le filtre :

- laisser passer l'eau
- retenir les particules du sol de base (ici, le noyau)

La charge trop élevée dans le noyau peut avoir plusieurs causes : soit le noyau, soit le filtre, ne joue pas bien un de ses rôles.

- le noyau ne retient pas assez l'eau

car il est fissuré partiellement dans sa largeur

- le filtre ne laisse pas passer assez d'eau

car il ne vérifie pas bien les critères de conception.

Afin de confirmer une des hypothèses émises, les experts (et plus tard le système expert) effectuent des tests complémentaires, étudient l'historique du comportement du barrage, observent d'autres mesures, vérifient les critères de conception etc... La vérification des critères de filtre est une opération réalisée actuellement par le système. C'est ce que prouve l'exemple développé ci-après.

Les critères que Sherard a énoncés en 1985 sont utilisés. Ils sont représentés par des règles, *données* du système expert. Il est bien sûr possible (c'est le but principal du système expert) d'utiliser n'importe quels autres critères, de Sherard à d'autres moments, ou d'autres experts. On peut même représenter tous les critères que l'on veut en même temps dans le système, afin par exemple d'en avoir une vue plus équilibrée et de ne pas se sentir attaché à l'avis d'un seul expert à un moment précis.

3. objets de base

Les critères de filtre de Sherard s'appliquent à des sols et des filtres dont on n'a besoin de connaître qu'un petit nombre de caractéristiques. Nous pouvons représenter les sols et les filtres comme des instances de classes du langage RLA créées à cette occasion. Nous pourrons alors en créer autant d'instances que de sols et de filtres que nous voudrons tester.

Voici ces classes :

```
objet              : sol
type               : classe
propriétés
#:system:loaded-from-file : barmot1.ll
Unité
définition         : les sols
sous-types         : ()
champs             : (d-85 %-passant-le-tamis-4
                      %-passant-le-tamis-200)
instances          : (noyau3)
Type
sélecteurs         : (%-passant-le-tamis-200
                      %-passant-le-tamis-4 d-85)


objet              : filtre
type               : classe
propriétés
#:system:loaded-from-file : barmot1.ll
Unité
définition         : les filtres
sous-types         : ()
champs             : (grand-d-15)
instances          : (filtre1)
type
sélecteurs         : (grand-d-15)
```

Voici comment créer quelques instances, objets de RLA, et donc objets faisant partie de notre base de données et accessibles directement au moteur :

```
[sol crée noyau3]          ; création de l'objet noyau3
[noyau3 <- d-85 5]         ; affectations des valeurs
[noyau3 <- %-passant-le-tamis-200 15]
[noyau3 <- %-passant-le-tamis-4 70]
```

```
[filtre crée filtre1]
[filtre1 <- grand-d-15 2.1]

[filtre crée filtre2]
[filtre2 <- grand-d-15 18]
```

L'objet noyau3, par exemple, existe donc maintenant et est représenté par (cette description est obtenue par l'envoi du message : `[noyau3 ??]`) :

```
objet              : noyau3
type               : sol
Unité
d-85               : 5
%-passant-le-tamis-4       : 70
%-passant-le-tamis-200     : 15
```

Comme nous l'avons vu, il est possible aussi de créer des faits sur des objets indépendamment des objets eux-mêmes. Créons ici des faits sur les objets noyaux1, noyau2 et `filtre1` :

```
[fait crée () (le d-85 de noyau1 est 2) ; l'énoncé
              (0.9 1)                   ; l'incertitude
              la                        ; la source
              #.(date)]                 ; la date
=fait7          ; réponse du système qui a créé le fait et
                ; l'a baptisé fait7

[fait crée () (le %-passant-le-tamis-4 de noyau1 est 95)
              (0.8 1)
              la
              #.(date)]
=fait8
```

```
[fait crée () (le %-passant-le-tamis-200 de noyau1 est 30)
              (0.8 1)
              la
              #.(date)]
=fait9
```

Une fois créés, il faut ajouter ces faits à la base de faits du moteur utilisé (qui s'appelle `m1`) :

```
[#.[m1 ? base-de-faits] ajoute fait7]

[#.[m1 ? base-de-faits] ajoute fait8]

[#.[m1 ? base-de-faits] ajoute fait9]
```

D'autres faits et objets sont créés à partir de fichiers (au format ASCII, i.e. définissables par l'utilisateur grâce à n'importe quel éditeur de texte).

4. règles de classement des sols

Préalablement à la vérification des critères, les sols sont classés dans quatre groupes. De même que l'on pourra vérifier les critères de Sherard, on peut bien sûr déterminer à quel groupe appartient un sol quelconque. Les définitions de ces groupes sont données par Sherard aux mêmes pages.

Voici les règles telles qu'elles sont représentées dans notre système :

```
objet            : règle6
type             : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition       : définition du groupe 1 (silt fin et
                   argile)
prémisses        : ((le %-passant-le-tamis-200 de Sol est
                     P%)
                    (vérifie (>= P% 85)))
conclusions      : (((Sol est du groupe 1) (1 1)))
source           : sherard
date             : 1 janv 88 00:00:00
variables        : (Sol P%)


objet            : règle7
type             : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition       : définition du groupe 2
prémisses        : ((le %-passant-le-tamis-200 de Sol est
                     P%200)
                    (le %-passant-le-tamis-4 de Sol est P%4)
                    (vérifie (and (<= 40 (* P%4 0.01 P%200))
                                  (< (* P%4 0.01 P%200)
                                     85))))
conclusions      : (((Sol est du groupe 2) (1 1)))
source           : sherard
date             : 1 jan 88 00:00:00
variables        : (Sol P%200 P%4)
```

```
objet           : règle8
type            : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition      : définition du groupe 3
prémisses       : ((le %-passant-le-tamis-200 de Sol est
                    P%200)
                   (le %-passant-le-tamis-4 de Sol est P%4)
                   (vérifie (<= (* P%4 0.01 P%200) 15)))
conclusions     : (((Sol est du groupe 3) (1 1)))
source          : sherard
date            : 1 janv 88 00:00:00
variables       : {Sol P%200 P%4}


objet           : règle9
type            : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition      : définition du groupe 4
prémisses       : ((le %-passant-le-tamis-200 de Sol est
                    P%200)
                  (le %-passant-le-tamis-4 de Sol est P%4)
                  (vérifie (and (<= 15 (* P%4 0.01 P%200))
                                (<= (* P%4 0.01 P%200)
                                    40))))
conclusions     : (((Sol est du groupe 4) (1 1)))
source          : sherard
date            : 1 janv 88 00:00:00
variables       : {Sol P%200 P%4}
```

Chaque règle est une instance de la classe des règles, définie comme suit :

```
objet           : règle
type            : classe-a
propriétés
#:system:loaded-from-file : motfr.ll
Unité
définition      : les règles utilisables avec le moteur
                  d'inférences
sous-types      : {}
champs          : (définition prémisses conclusions source
                   date variables)
instances       : {règle12 règle11 règle10 règle9 règle8
                   règle7 règle6 règle5 règle4 règle3
                   règle2 règle1}
Type
sélecteurs      : {variables date source conclusions
prémisses définition post-supprime-annule-conclusions post-
ajoute-met-conclusions post-annule-conclusions post-
supprime-annule-prémisses post-ajoute-met-prémisses post-
annule-prémisses si-besoin-variables}
```

Dans notre exemple, douze règles existent : de `règle1` à `règle12`. Ces douze règles composent la base de règles du moteur `m1` que nous utilisons. Voici le sens de chacune de ces règles :

`règle1` : vérifie le critère de Sherard à partir du D_{15} maximum autorisé pour le filtre

`règle2` à `règle4` : définissent le D_{15} maximum autorisé pour le filtre à partir du groupe du sol de base

`règle6` à `règle9` : déterminent le groupe d'un sol.

Les trois autres règles (`règle10` à `règle12`) donnent au moteur la possibilité d'accéder aux attributs d'un objet quand il en a besoin, et à effectuer des calculs ou des vérifications de tests booléens. Ce sont des règles du domaine de l'informatique, en opposition aux règles `règle1` à `règle9` qui sont du domaine de la géotechnique. Leur explication sort du cadre de ce mémoire; elle ferait appel à des connaissances précises et détaillées du fonctionnement du moteur d'inférences. En cas d'utilisation d'autres critères de filtre, seules les règles `règle1` à `règle9` seraient à modifier, voire à détruire et à remplacer par d'autres.

5. classement des sols

Utilisons maintenant le moteur (`m1`) pour déterminer à quelles catégories appartiennent les sols que nous avons définis. Pour cela, nous lui posons des questions.

- noyau3 est-il du groupe 3?

question : `[m1 réponds (noyau3 est du groupe 3)]`

Il n'y a pas ici de variable dans notre question. Le moteur se contente de trouver quelle certitude il peut accorder au fait que nous lui proposons.

réponse du moteur : `() (1 1) fait22`

Le moteur a démontré que "noyau3 est du groupe 3" est un fait certain : la certitude qu'il donne est le couple (1 1). Le moteur a également crée automatiquement un fait (l'objet `fait22`) qui conserve le résultat de cette recherche, avec sa justification.

♦ De quel groupe est noyau1?

question : `[m1 réponds (noyau1 est du groupe Groupe)]`

Remarquez ici la présence de la variable `Groupe` (symbole qui commence par une majuscule) dans la question. Cela signifie que l'on demande au moteur de nous donner la ou les valeurs possibles pour cette variable.

réponse du moteur : `(Groupe . 4) (0.8 1) fait15`

La seule valeur que le moteur a trouvée pour remplacer la variable `Groupe` est 2. Le noyau2 est donc du groupe 2. De plus, c'est un fait seulement presque certain : l'incertitude donnée par le moteur est (0.8 1), i.e. presque complètement nécessaire (0.8), et complètement possible (1). Le fait fait15 conserve ce résultat et sa justification, et reste disponible pour toute consultation ultérieure.

justification détaillée. Nous demandons l'affichage de l'arbre de déduction :

```
----> ((G . 4)) (0.8 1) fait15
mpg (0.8 1) (noyau1 est du groupe 4)
├──règle9 définition du groupe 4
├──mpg (1 1) (vérifie (and (<= 15 (* 95 100 30))
│   │                        (<= (* 95 100 30) 40)))
│   ├──règle11 vérifie un test booléen
│   └──évalue
├──fait8 (0.8 1) (le %-passant-le-tamis-4 de noyau1 est 95)
└──fait9 (0.8 1) (le %-passant-le-tamis-200 de noyau1 est
                  30)
```

Le moteur a effectué deux Modus Ponens généralisés. Il a utilisé une fois la règle `règle9` qui définit les sols du groupe 4, une fois la règle `règle11` qui permet la vérification des tests booléens. Il a aussi utilisé les faits de la base `fait8` et `fait9`, et a combiné leurs incertitudes. Toutes les valeurs qu'il a utilisées apparaissent en clair dans la justification.

♦ De quel groupe est noyau2?

question : `[m1 réponds (noyau2 est du groupe G)]`

réponses :

```
(G . 2) (0.5 1) fait12
(G . 3) (0 0.6) fait4
```

Le moteur donne ici deux réponses : il est assez probable (0.5 1) que noyau2 soit du groupe 2, et il est plutôt certain qu'il ne soit pas (0 0.6) du groupe 3.

justification détaillée :

```
----> ((G . 2)) (0.5 1) fait12
mpg (0.5 1) (noyau2 est du groupe 2)
├──règle7 définition du groupe 2
├──mpg (1 1) (vérifie (and (<= 40 (* 80 100 60))
│   │                        (< (* 80 100 60) 85)))
│   ├──règle11 vérifie un test booléen
│   └──évalue
├──fait2 (0.5 1) (le %-passant-le-tamis-4 de noyau2 est 80)
└──fait3 (0.7 1) (le %-passant-le-tamis-200 de noyau2 est
                  60)

----> ((G . 3)) (0 0.6) fait4
fait4 (0 0.6) (noyau2 est du groupe 3)
```

La première réponse est le fruit de deux applications de règles, d'utilisation de faits de la base, et de combinaison des incertitudes. La deuxième est simplement le rappel du fait déjà présent dans la base.

Notons que, sans leur incertitude, ces deux faits seraient contradictoires (car la partition des sols en quatre groupes est disjointe). Mais la présence des incertitudes permet au système les traiter en même temps sans problème.

6. critères de Sherard

Les quatre critères de Sherard sont représentés aussi par des règles dans notre système expert. Ces règles ont été écrites *facilement* à partir de l'article de Sherard, grâce à l'expressivité du langage RLA, à la présence de variables, et, bien sûr, à la clarté de leur énoncé dans l'article. On s'en persuadera en regardant leurs prémisses et leurs conclusions.

Ces règles sont toutes basées sur le même principe : on établit d'abord un D_{15} maximum possible pour le filtre, en fonction du sol en question; puis on vérifie que le D_{15} réel du filtre est bien inférieur à ce maximum. Si c'est le cas, on affirme que le critère est très probablement (0.9 1) vérifié.

Voici, montrés en parallèle, les critères de Sherard et les règles qui les représentent :

- règles qui définissent le D_{15} maximum en fonction du sol considéré :

critère de Sherard : le D_{15} maximum autorisé pour un filtre qui retient un sol du groupe 1 est $9 \times d_{85}$.

sa représentation au moyen d'une règle :

```
objet           : règle2
type            : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition      : grand-d-15 maxi pour les sols du groupe 1
prémisses       : ((Sol est du groupe 1)
                    (1 1)))
                   (le d-85 de Sol est D-85)
                   (Grand-d-15-maxi vaut (* 9 D-85)))
conclusions     : (((le grand-d-15 maxi du sol Sol est
                     Grand-d-15-maxi)
source          : sherard
date            : 1 janv 88 00:00:00
variables       : (Sol D-85)
```

critère de Sherard : le D_{15} maximum autorisé pour un filtre qui retient un sol du groupe 2 est 0.7mm.

sa représentation au moyen d'une règle :

```
objet           : règle3
type            : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition      : grand-d-15 maxi pour les sols du groupe 2
prémisses       : ((Sol est du groupe 2))
conclusions     : (((le grand-d-15 maxi du sol Sol est 0.7)
                    (1 1)))
source          : sherard
date            : 1 janv 88 00:00:00
variables       : {Sol}
```

critère de Sherard : le D_{15} maximum autorisé pour un filtre qui retient un sol du groupe 3 est $4 \times d_{85}$.

sa représentation au moyen d'une règle :

```
objet           : règle4
type            : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition      : grand-d-15 maxi pour les sols du groupe 3
prémisses       : ((Sol est du groupe 3)
                   (le d-85 de Sol est D-85)
                   (Grand-d-15-maxi vaut (* 4 D-85)))
conclusions     : (((le grand-d-15 maxi du sol Sol est
                      Grand-d-15-maxi)
                     (1 1)))
source          : sherard
date            : 1 janv 88 00:00:00
variables       : {Sol D-85 Grand-d-15-maxi}
```

critère de Sherard : le D_{15} maximum autorisé pour un filtre qui retient un sol du groupe 4 est $0.7 + (40 - \%_{\#200})/25 \times (4 \times d_{85} - 0.7)$.

sa représentation au moyen d'une règle :

```
objet              : règle5
type               : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition         : grand-d-15 maxi pour les sols du groupe 4
prémisses          : ((Sol est du groupe 4)
                      (le d-85 de Sol est D-85)
                      (le %-passant-le-tamis-200 de Sol est
                       P%200)
                      (Grand-d-15-maxi vaut
                        (+ 0.7
                           (* (/ (- 40 P%200) 25)
                              (- (* 4 D-85) 0.7)))))
conclusions        : (((le grand-d-15 maxi du sol Sol est
                        Grand-d-15-maxi)
                       (1 1)))
source             : sherard
date               : 1 janv 88 00:00:00
variables          : {Sol D-85 P%200 Grand-d-15-maxi}
```

- La règle unique qui, à partir du D_{15} maximum autorisé pour le filtre déterminé à partir des caractéristiques du sol de base, vérifie le critère est la suivante :

```
objet            : règle1
type             : règle
propriétés
#:system:loaded-from-file : barr2.ll
Unité
définition       : critère de filtre pour tous les sols
prémisses        : ((le grand-d-15 maxi du sol Sol est
                     Grand-d-15-maxi)
                    (le grand-d-15 de Filtre est Grand-d-15)
                    (vérifie (<= Grand-d-15
                                 Grand-d-15-maxi)))
conclusions      : (((critère ok pour le sol Sol et le
                     filtre Filtre)
                    (0.9 1)))
source           : sherard
date             : 1 janv 88 00:00:00
variables        : {Sol Grand-d-15-maxi Filtre Grand-d-15}
```

Cette règle se contente de vérifier la relation : $D_{15} \leq D_{15maxi}$.

7. vérification des critères

Vérifier les critères de Sherard sur les sols est maintenant chose facile. Notre moteur disposant de tous les éléments, il suffit de lui poser les bonnes questions.

♦ Quel est le D_{15} maximum d'un filtre pour le sol noyau2?

```
question : [m1 réponds (le grand-d-15 maxi du sol noyau2 est
    G-d-15-maxi)]
```

réponse : `(G-d-15-maxi . 0.7) (0.5 1) fait29`

La valeur est 0.7 mm; c'est assez probable (0.5 1).

♦ Le critère est-il vérifié pour le noyau noyau2 et le filtre `filtre2`?

question : `[m1 réponds (critère ok pour le sol noyau2 et le filtre filtre2]`

réponse : `()` : non.

♦ Le critère est-il vérifié pour le noyau noyau1 et le filtre `filtre1`?

question : `[m1 réponds (critère ok pour le sol noyau1 et le filtre filtre1)]`

réponse : `() (0.8 1) fait59`

oui; c'est pratiquement sûr (0.8 1).

justification détaillée : (arbre de déduction)

```
mpg (0.8 1) (critère ok pour le sol noyau1 et le filtre
             filtre1)
├──règle1 critère de filtre pour tous les sols
├──mpg (1 1) (vérifie (<= 2.1 3.62))
│   ├──règle11 vérifie un test booléen
│   └──évalue
├──mpg (1 1) (le grand-d-15 de filtre1 est 2.1)
│   ├──règle10 accès à l'attribut d'un objet
│   └──évalue
└──mpg (0.8 1) (le grand-d-15 maxi du sol noyau1 est 3.62)
    ├──règle5 grand-d-15 maxi pour les sols du groupe 4
    ├──mpg (1 1) (3.62 vaut (+ 0.7 (* (/ (- 40 30) 25)
    │                                 (- (* 4 2) 0.7))))
    │   ├──règle12 calcule une valeur à partir d'une formule
    │   └──évalue
    ├──fait9 (0.8 1) (le %-passant-le-tamis-200 de noyau1
    │                 est 30)
    ├──fait7 (0.9 1) (le d-85 de noyau1 est 2)
    └──mpg (0.8 1) (noyau1 est du groupe 4)
        ├──règle9 définition du groupe 4
        ├──mpg (1 1) (vérifie (and (<= 15 (* 95 100 30))
        │                          (<= (* 95 100 30) 40)))
        │   ├──règle11 vérifie un test booléen
        │   └──évalue
        ├──fait8 (0.8 1) (le %-passant-le-tamis-4 de noyau1
        │                 est 95)
        └──fait9 (0.8 1) (le %-passant-le-tamis-200 de
                          noyau1 est 30)
```

Le moteur s'est appelé de façon récursive (i.e. lui-même) plusieurs fois et allant jusqu'à une profondeur de quatre niveaux. L'utilisateur peut étudier la justification dans les détails qu'il souhaite, il ne lui est pas nécessaire de "s'enfoncer" dans toutes les profondeurs du raisonnement. (Une interface conversationnelle, qui reste à écrire, doit permettre à l'utilisateur de n'afficher que ce dont il a besoin.)

Quel est ce fait59 créé?

```
objet           : fait59
type            : fait
Unité
énoncé          : (critère ok pour le sol noyau1 et le
                   filtre filtre1)
certitude       : (0.8 1)
source          : (une longue liste qui représente l'arbre de
                   déduction)
date            : lun 13 nov  89 17:22:25
```

- Le noyau noyau2 et le filtre filtre1 vérifient-ils le critère de Sherard?

question : [m1 réponds (critère ok pour le sol noyau2 et le filtre filtre1)]

réponse : () (0 0.6) fait41

Il est assez improbable (0 0.6) que le critère soit vérifié pour le noyau2 et le filtre1.

D. détection et explication des anomalies par le système expert

1. rappel de la problématique

Le but ultime de notre système expert est la détection et l'explication des mauvais fonctionnements éventuels des barrages en terre. Nous disposons d'un langage nous permettant de représenter et d'accéder à toutes les connaissances pour la résolution d'un tel problème, et dans lequel nous avons pu écrire un moteur d'inférences. L'exemple donné

précédemment montre que le système fonctionne; mais les bases de connaissances utilisées sont encore bien trop simples pour parler d'un système expert industriel.

Résoudre le problème général initial nécessite préalablement le développement de bases de connaissances complexes et complètes. Les connaissances impliquées sont autant du domaine des faits que du domaine des méthodes de raisonnement. Montrons quel peut être un schéma de raisonnement des experts confrontés au problème, et comment construire les éléments de sa représentation et de sa reproduction par le système expert.

2. schéma de raisonnement

D'après quelques observations et dialogues avec des experts, il est possible de mettre en évidence quelques points clés dans le processus de détection et d'explication des anomalies des barrages en terre.

a) monde supposé

L'expert possède une *représentation* du monde réel : c'est son monde supposé. (On entend par "monde" l'ensemble des éléments pertinents pour le problème en question.) Ses connaissances du monde peuvent être plus ou moins précises et plus ou moins certaines. Il est possible que la représentation ne soit pas fidèle, et que des différences et des erreurs existent. Le raisonnement doit, soit ne pas être bloqué par de telles erreurs, soit permettre leur découverte et leur correction.

L'expert possède aussi, comme on l'a déjà vu, un grand nombre de connaissances méthodologiques de plusieurs origines : livresques, de bon sens ou acquises par son expérience.

La représentation du monde de notre système expert, que des connaissances de méthodes et du savoir-faire, est constituée par l'ensemble des diverses bases de données, de faits, de règles et de méta-règles. Plusieurs bases de natures différentes peuvent coexister, et apparaître et disparaître au fil des raisonnements.

b) hypothèses et mondes hypothétiques

A partir de observations, mesures et autres faits d'une part, et des connaissances qui caractérisent les bons fonctionnements des barrages d'autre part, l'expert est apte à déterminer si il y a des anomalies dans le fonctionnement d'un barrage particulier. Ayant détecté des anomalies éventuelles, il cherchera à les expliquer en émettant plusieurs hypothèses. La construction de ces hypothèses semble être principalement le fruit de son expérience.

Pour chaque hypothèse émise ou chaque combinaison d'hypothèses, l'expert développe un monde dans lequel il met à jour ses connaissances, déduit de nouveaux faits, propage les effets des faits supposés. Ce processus conduit l'expert à avoir simultanément en tête plusieurs mondes hypothétiques.

Le système expert doit posséder les connaissances empiriques de l'expert afin de pouvoir proposer les mêmes hypothèses. C'est une des

principales difficultés de l'écriture des bases de connaissances du système expert. Les connaissances empiriques sont difficiles d'accès et demandent une collaboration étroite et longue entre l'expert et le "cogniticien" chargé de représenter les connaissances pour le système expert.

c) raisonnement : développements sélectifs

Les mondes hypothétiques sont classés par l'expert de façon plus ou moins consciente, du plus probable ou vraisemblable, au plus improbable ou étrange. Ce classement est, lui aussi, basé sur les connaissances de l'expert : les connaissances qui caractérisent pour lui un monde "normal". Certaines situations sont "normales", d'autres probables, d'autres fort improbables. Plus l'expert a d'expérience dans le domaine, plus ce classement lui est facile. A partir de ce classement, l'expert peut éliminer les mondes hypothétiques les moins probables. Il peut ensuite répéter l'étape précédente : émettre des hypothèses à partir de chaque monde hypothétique restant. Ce processus donne naissance à plusieurs nouveaux mondes hypothétiques, et peut se répéter plusieurs fois de suite. On voit bien la nécessité de classer les mondes et d'éliminer les mondes les moins probables, sans quoi le nombre de mondes hypothétiques croîtrait en saturant la mémoire et le raisonnement.

Le système expert peut reproduire un tel processus. A partir d'un ensemble de mondes hypothétiques à un moment donné, il doit les classer suivant leur vraisemblance et éliminer les plus invraisemblables. Pour ce faire, il doit disposer des connaissances caractérisant les mondes "normaux". Ces connaissances sont données par son expérience, bien sûr, et

aussi par les lois de la physique, de l'hydraulique et de la géotechnique. Une fois construite, la liste ordonnée des mondes vraisemblables restants sert de base à une nouvelle application du processus, et à la construction de nouvelles listes de mondes hypothétiques.

d) diagnostic

L'expert, humain, est limité par sa mémoire et ne peut se représenter un grand nombre de mondes hypothétiques simultanément. Pourtant il peut fournir les analyses et explications des problèmes. Ce qui est important pour son raisonnement est la précision et la justesse avec laquelle il crée et évalue la vraisemblance des mondes hypothétiques, et sa capacité à réduire le problème en ne conservant que les mondes dignes d'intérêt. Au bout de quelques hypothèses, un ou plusieurs mondes hypothétique très vraisemblables peuvent apparaître. Le processus de développement en chaîne s'arrête. Les diagnostics proposés par l'expert sont les différents ensembles d'hypothèses qui ont abouti aux mondes les plus vraisemblables.

En mettant à la disposition du système expert des connaissances assez précises et justes pour l'évaluation de la vraisemblance des mondes, on peut s'attendre à ce que le système produise aussi un ensemble de diagnostics plausibles, et ce dans un temps acceptable. Pour cela, il faut utiliser au maximum les méta-connaissances permettant au systèmes de faire les choix les plus judicieux, et le rendant ainsi efficace.

A partir de l'ensemble de diagnostics d'une anomalie, des décisions de divers ordres peuvent être prises : ne rien faire (si l'anomalie et ses causes ne sont pas importantes), prendre des mesures simples de correction,

envisager des travaux plus importants, voire déclencher des mesures d'urgence... Cette prise de décision quant à ce qu'il faut faire en cas d'une anomalie, disposant de son diagnostic, est une tâche qui sort des buts fixés pour notre système expert, mais que l'on peut envisager comme un prochain défi pour l'intelligence artificielle appliquée au domaine des barrages.

conclusion

rappel des objectifs

Le but de la recherche engagée dans cette maîtrise est l'élaboration d'un système expert pour l'évaluation des barrages en terre, la détection des anomalies éventuelles, leur analyse et leur explication. Pour ce genre de problèmes, l'informatique traditionnelle est limitée fondamentalement et ne permet pas de répondre à toutes nos exigences; l'intelligence artificielle est utilisée car c'est une approche alternative qui permet de dépasser ces limites en tentant de reproduire les comportements humains dans les tâches invoquées. La réalisation du système expert nécessite plusieurs grandes étapes: inventaire et modélisation des multiples connaissances mises en oeuvre, écriture d'un moteur d'inférence capable d'utiliser ces connaissances, élaboration de stratégies de résolution de problèmes de barrages calquées sur celles des experts.

réalisations : objectifs atteints

Les études bibliographiques et recherches entreprises ont conduit à la réalisation de plusieurs pièces maîtresses pour le système expert.

- Les connaissances impliquées dans la résolution du problème par l'homme on été mises en évidence, et une philosophie de représentation a été proposée pour répondre aux attentes du système expert.

- Un langage de représentation de ces connaissances a été développé, qui permet aussi l'implantation des algorithmes; ce langage combine des caractéristiques qui en font un langage adapté au développement des systèmes experts en général.
- Un moteur d'inférences original a été écrit. C'est un moteur d'ordre 1 qui fonctionne en chaînage arrière, qui permet l'utilisation de bases de connaissances très diverses, la prise en compte et le traitement des connaissances incertaines, l'utilisation (future) de méta-connaissances.

Le système expert fonctionne avec de petites bases de connaissances pour des cas simples; des approches ont été proposées pour augmenter la taille des bases, compliquer les raisonnements, et ainsi se rapprocher des résultats des experts.

comment perfectionner le système

Des nombreuses améliorations sont possibles pour le système expert. Sa version actuelle est un prototype : il n'est pas optimisé, pas compilé (bien que ceci soit possible à très peu de frais), et en évolution permanente.

Le langage développé pour la réalisation de notre système expert, RLA, peut être vu comme constitué de plusieurs couches d'objets de plus en plus spécialisées. Son évolution se fait par ajouts de nouvelles couches qui sont autant d'outils mis à la disposition du programmeur, au fur et à mesure des utilisations. Par exemple, des outils sont souhaitables très rapidement pour l'utilisation des fenêtres, du "multi-tâches", ou pour améliorer l'interface entre les différents utilisateurs et le système.

La possibilité de représenter des informations incertaines est présente, mais demeure insuffisante pour représenter l'"incomplétude" de notre savoir. Il est nécessaire dans les développements proches du système de permettre aussi la représentation des connaissances *imprécises*.

Les méta-connaissances se révèlent être les moyens indispensables de l'efficacité du système expert dans les problèmes réels de grande envergure. Tout est actuellement en place pour les utiliser, à quelques adaptations près du moteur (où les faire intervenir? comment éviter les bouclages et les ascensions de niveaux infinies?); la difficulté fondamentale n'est pas là. Elle réside dans l'énoncé des méta-connaissances, dans leur élaboration et leur organisation. Ce problème est aujourd'hui un des domaines principaux de la recherche en intelligence artificielle, et auquel il est souhaitable de participer à l'occasion de la réalisation d'un système expert comme le notre.

En résumé, le système écrit est effectif, mais il est nécessaire d'y consacrer encore beaucoup d'efforts pour le rendre "industriel", i.e. utilisable en grandeur nature.

applications à d'autres domaines

Les principes utilisés pour la réalisation du système expert sont, pour la plupart, généraux et indépendants du domaine particulier des barrages en terre. Il en va de même pour certaines parties des bases de connaissances. Des résultats de recherches, tant fondamentaux et concernant l'intelligence artificielle, que spécialisés dans des domaines divers, voire éloignés du domaine des barrages, ont été utilisés. Le système

et les résultats auxquels nous avons abouti peuvent être utilisés en génie civil, en génie en général, et dans de nombreux autres domaines. Les capacités qui font l'intelligence humaine peuvent s'appliquer à une immense diversité de problèmes. La vocation du système expert est de reproduire ces capacités et de profiter de leur universalité.

références bibliographiques

[AVON 88] *Un environnement pour l'étude de la logique temporelle*
Lionel AVON, rapport de D.E.A. en Intelligence Artificielle et Applications, Université de Caen, France, 1988

[ASCE 87] *Expert Systems for Civil Engineers : Technology and Application*
A.S.C.E., 1987

[CAMARERO 88] *L'intégration en conception assistée par ordinateur pour les projets pluridisciplinaires de génie civil*
R. Camarero, L. Granger, C. Marche, M. Soulié et R. Tinawi, Revue canadienne de génie civil, vol. 15, n. 6, 1988

[CHATAIN 87] *Systèmes experts, méthodes et outils*
Jean-Noel Chatain, Alain Dussauchoy, EYROLLES, 1987

[CIGB 82] *L'automatisation dans le contrôle de la sécurité des barrages*
C.I.G.B., bulletin 41, 1982

[DAVEY 89] *Development of a Knowledge-based System for the Selection of Groundwater Control Methods*
I.E.G. Davey-Wilson, I.M. May, Computer and Geotechnics 7, 1989, 189-203

[DAVIS 80a] *Meta-rules : Reasoning about Control*
Artificial Intelligence 15, 1980, 179-222

[DAVIS 80b] *Content Reference : Reasoning about Rules*
Artificial Intelligence 15, 1980, 223-239

[DETIENNE 88] *Une revue des études psychologiques sur la compréhension des programmes informatiques*
Françoise Détienne, Technique et Science Informatiques, vol. 8, no. 1, 1989, 5-20

[DUBOIS 85] *Théorie des possibilités*
D. Dubois, H. Prade, Masson, 1985

[DUCOURNEAU 89] *La multiplicité de l'héritage dans les langages à objets*
Roland Ducourneau, Michel Habib, Technique et Science Informatiques, vol. 8, no. 1, 1989, 41-62

[FOUET 85] *La machine GOSSEYN*
Jean-Marc Fouet, Publications du Laboratoire C.F. Picard (C.N.R.S.), Colloque d'Intelligence Artificielle, Toulouse, France, 1985, 203-222

[GALLANTI 85] *Representing Procedural Knowledge in Expert Systems : an Application to Process Control*
Massimo Gallanti et al., 9th. I.J.C.A.I. (International Joint Conference on Artificial Intelligence), 1985, 345-352

[HAASE 84] *ARLO (Another Representation Language Offer), the Implementation of a Language for Describing Representation Languages*
Kenneth W. Haase, M.I.T., 1984

[HYDRO 82] *Comportement de la digue du réservoir lac Sainte-Anne*
Guy Saint-Arnaud, Service géologie et géotechnique, Hydro-Québec, 35ième Conférence Canadienne de Géotechnique, Montréal, septembre 1982

[HYDRO 84] *Installation des puits de décompression à la digue sud-est du réservoir lac Sainte-Anne*
Hydro-Québec, Direction ingénierie des centrales, Service géologie et géotechnique, 1984

[HYDRO 86] *Comportement de la digue sud-est du réservoir lac Sainte-Anne*
Hydro-Québec, Direction des équipements de production, Service géologie et géotechnique, 1986

[INRIA 87] *Le_Lisp, version 15.21, manuel de référence*
I.N.R.I.A., France, 1987

[LESSARD 89] *Validation du système de gestion des données d'auscultation des barrages de la région Manicouagan*
Ghislain Lessard, Castorplus, projet CDT 1336, CDT 02/1989

[LIUC 88a] *AIRELLE, manuel d'utilisation*
Anne Adam-Nicolle et al., Laboratoire d'Informatique de l'Université de Caen, France, 1988

[LIUC 88b] *AIRELLE, rapport de présentation*
Anne Adam-Nicolle et al., Laboratoire d'Informatique de l'Université de Caen, France, 1988

[MARRIOTT 89] *On PROLOG and the Occur Check Problem*
Kim Marriott, Harald Sondergaard, Sigplan notices, vol. 24, no. 5, 1989, 76-82

[MARTIN 85a] *SPII-1 : un moteur d'inférences capables de traiter des informations imprécises ou incertaines*
Roger Martin-Clouaire, Henri Prade, COGNITIVA 85, Paris, France, 4-7 juin 1985

[MARTIN 85b] *Connaissances et méta-connaissances dans ELFIN*
Roger Martin-Clouaire, Colloque d'Intelligence Artificielle, Toulouse, France, 16-20 septembre 1985, 123-149

[PITRAT 85] *Utilisation des connaissances déclaratives*
Jacques Pitrat, Ecole Internationale d'Informatique de l'AFCET, Aix-en-Provence, France, Juillet 1985

[ROCHE 89] *Les approches objets et le langage LRO2 (KEOPS)*
Christophe Roche, Jean-Pierre Laurent, Technique et science Informatiques, vol. 8, no. 1, 1989

[SEBJ 85a] *Rapport d'étape sur le comportement des ouvrages et structures (période d'avril 1984 à avril 1985), aménagement LG3*
S.E.B.J. 05/1985

[SEBJ 85b] *LG3 Behavior Analysis of Dike TA-26B*
S.E.B.J. 08/1985

[SHERARD 85] *Filters and Leakage Control in Embankments Dams*
James L. Sherard, F. A.S.C.E., Lorn P. Dunnigan, M. A.S.C.E., 1985, pp. 1-30

[SMITH 87] *Exposé sur l'informatisation des observations visuelles des ouvrages en terre de la Baie James*
Marc Smith, Ouvrages de génie civil, Région de La Grande, 1988 (rapport interne, Hydro-Québec)

[WILSON 79] *Current Trends in Design and Construction of Embankment Dams*
Stanley D. Wilson, Raul J. Marsal, A.S.C.E. 1979

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00290820 8