**MASTER EN SCIENCES INFORMATIQUES**

# Conception d'un système d'aide à la mémorisation

Cayphas, Philippe

*Award date:*
1988

*Awarding institution:*
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur

Intitut d'Informatique

Année académique 1987 - 1988

# Conception d'un système d'aide à la mémorisation

Philippe CAYPHAS

Mémoire présenté en vue de l'obtention du titre de licencié et maître en informatique

Promoteur : C. CHERTON

Nous remercions tout particulièrement Mr Cherton, promoteur de ce mémoire, pour ses précieux conseils, sa très grande patience et son entière disponibilité.

Nous adressons également nos sentiments de profonde reconnaissance à tous ceux qui nous ont aidé à mener à bien ce travail.

## ABSTRACT

This university thesis deals with computers assisted teaching. It tries to show that it is now possible to develop something quite different from traditionnal didactic programs.

Thanks to the system of memorization we propose, the teacher can describe a relationnaly structured knowledge and different parametrized rules (meta-knowledge).

PLAN

# INTRODUCTION GENERALE

L'apparition de l'informatique dans notre société est la cause de nombreux changements dans notre vie quotidienne.

L'enseignement est également le théatre de ces modifications. Toutefois, bien qu'il s'agisse d'un domaine très vaste, il n'existe actuellement que des didacticiels très spécialisés et bien souvent peu intéressants pour les enseignants.

Nous estimons que, dans ce domaine, beaucoup reste à faire. L'objectif de ce travail est précisément de montrer qu'il est possible d'imaginer un système plus général n'ayant pas le caractère limité des didacticiels.

Contrairement à ce qui existe déjà sur le marché scolaire, le système que nous proposons présente l'originalité suivante : il permet au professeur de décrire synthétiquement, d'une part, la matière, et d'autre part, une série de règles qui permettent au système de fonctionner en aidant judicieusement l'étudiant.

Nous avons choisi de concevoir un système d'aide à la mémorisation car cet aspect de l'apprentissage est un problème qui n'est pas toujours facile à surmonter. L'apport de l'informatique peut, selon nous, faciliter l'approche de ce problème.

En effet, puisque la mémorisation met en oeuvre des processus non encore identifiés, son fonctionnement reste innexpliqué. Nous ne connaissons pas, non plus, le mode de représentation des informations dans la mémoire humaine. Toutefois, nous savons que celles-ci sont perçues par les cinq sens de l'homme.

Dès lors, présenter des informations à mémoriser en utilisant autant que possible les sensations ne peut qu'aider celui qui mémorise. C'est l'un des avantages qu'apporte l'informatique. En effet, n'est elle pas synonyme de "technique pour le traitement de l'information"?

De plus, l'informatique peut aussi aider l'étudiant dans des tâches plus traditionnelles : s'assurer que toutes la matière a bien été vue, par exemple.

Ne perdons quand même pas de vue que mémoriser est une action qui reste encore secrète. De plus, elle met en oeuvre de nombreux mécanismes. Nous en avons retenu un seul pour le développement du système: l'association..

Quand nous mémorisons une donnée, nous l'associons à une autre donnée, que nous connaissons déjà. Par exemple, nous retenons le numéro de téléphone de quelqu'un que nous connaissons.

Notre système sera uniquement prévu pour aider à ce genre de mémorisation. Il ne permettra pas de tenir compte de règles de déductions, d'algorithmes, ou de moyens mnémotechniques qui eux aussi pourtant, permettent de mémoriser.

Dans le premier chapitre, nous proposerons une synthèse de ce qu'ont apporté les sciences humaines dans le domaine de la mémorisation.

A partir de deux scénarios fictifs de mémorisation, nous relèverons, dans le deuxième chapitre, les points dont tiendrait compte un professeur qui aidant un étudiant qui mémoriserait des informations.

A la lumière des deux premiers chapitres, nous spécifierons le système d'aide à la mémorisation dans le troisième chapitre et en ferons une découpe modulaire dans le quatrième.

Dans le cinquième chapitre figurera l'implémentation d'une maquette utilisant les grandes caractéristiques définies auparavant dans le travail. Le sixième chapitre présentera les limites de notre travail.

# CHAPITRE I : LA MEMORISATION.

Etant donné notre objectif d'aide à la mémorisation, il est intéressant de voir ce que l'on connaît déjà sur cet aspect de l'apprentissage.

Dans ce premier chapitre, nous verrons ce que peuvent apporter les sciences humaines sur le fonctionnment de la mémoire. Nous en retiendrons, ensuite, les quelques propositions utiles au développement de notre système.

## I.1 L'apport des sciences humaines

Les recherches actuelles visant à comprendre le fonctionnement de la mémoire d'un être humain sont encore en plein développement; ancune théorie probante n'a encore pu voir le jour.

Certaines théories actuelles avancent toutefois l'hypothèse selon laquelle la mémoire se subdiviserait en mémoire immédiate (I.1.1), mémoire à court (I.1.2) et à long terme (I.1.3). Des recherches ont également été faites pour mieux cerner l'oubli (I.1.4).

Nous énumérerons en I.1.5 des conditions psychologiques favorables à une bonne mémorisation.

Outre les différentes mémoires vues ci-dessus,il existe également des propositions concernant la représentation des données mémorisées existent. Nous verrons, en I.1.6 ce que celles-ci peuvent apporter.

Toutes les idées développées dans ce chapitre sont référencées en 1 dans la bibliographie.

### I.1.1 la mémoire sensorielle (mémoire immédiate)

C'est elle qui permet de répèter ce qui vient d'être perçu. Elle dure quelques secondes seulement.

### I.1.2 la mémoire à court terme

C'est la mémoire de travail. Elle permet de retenir certaines sensations. On estime qu' elle peut retenir, en moyenne, sept éléments discrets pendant trente secondes.

### I.1.3 la mémoire à long terme

C'est dans celle-ci que sont stockées les informations que nous retiendrons plus longtemps. Elle demande un effort de la part de la personne qui mémorise afin d' attacher une nouvelle connaissance à toutes celles préexistantes.

C'est, évidemment, cette mémoire qui nous intéresse le plus dans le cadre de ce travail, les autres étant beaucoup trop volatiles et n'offrant donc que peu ou pas d'intérêt dans le cadre d'une mémorisation durable.

### I.1.4 L'oubli

Ce paragraphe présente quelques conclusions qui ont été tirées de nombreuses recherches sur la mémorisation (Référence 1).

**La mémoire à court terme renferme un nombre limité d'éléments bruts pendant une durée brève**

**La mémoire à long terme renferme un nombre immense d'informations pendant une période très longue.**

**L'oubli est sélectif, une sensation est d'autant mieux retenue qu'elle est reliée à d'autres.**

**L'oubli est croissant. Il est fonction du temps et du nombre d'éléments à mémoriser**

Ces affirmations pourraient toutefois être discutées.

### I.1.5 Conditions favorisant une bonne mémorisation.

- Revoir assez rapidement ce qui vient d'être mémorisé.
- Aménager des pauses pendant la mémorisation.
- Aménager, après l'apprentissage, des réactivations.
- Structurer les éléments à mémoriser.
- Utiliser les diverses formes de mémoire.

### I.1.6 Représentation des données en mémoire humaine.

Les sciences humaines n'apportent ici aussi que des éléments non démontrés. Par exemple, la métaphore du réseau sémantique est parfois utilisée pour expliquer le mode de représentation des données en mémoire. Toutefois, le lecteur remarquera clairement l'imprécision de la définition suivante :

" L'information serait stockée en mémoire sous forme de réseaux (non-linéaires), sémantiques où les noeuds seraient des objets, des événements, des actions, des concepts et des liaisons entre les noeuds des relations, des règles..."(1)

## I.2 Les sciences humaines et l'aide à la mémorisation.

Dans le développement du programme, on pourrait essayer de tenir compte des différents éléments vus plus haut, tout en sachant que rien n'est sûr à ce sujet et qu'il faut donc passer à l'application d'une théorie incertaine.

Malgré cette incertitude quant au fonctionnement de la mémoire et à la représentation qu'elle utilise, il existe des moyens facilitant la mémorisation: nous pouvons,par exemple, essayer de présenter à l'élève, l'information à mémoriser de façon à ce qu'il la retienne plus facilement. En effet, certains diront qu'ils ont une mémoire plutôt visuelle ou plutôt auditive. Il faut, dès lors, profiter de cet état de fait dans la façon de présenter les informations à mémoriser.

Nous pouvons également tenir compte d'éléments qui, même s'ils ne sont pas scientiquement démontrés, semblent néanmoins utiles : ne pas mémoriser des informations sans les structurer, par exemple.

Donc il semble malgré tout, possible, dans l'optique de notre système d'aide à la mémorisation, de garder quelques-uns des éléments apportés par les sciences de l'apprentissage. Nous en avons retenu quatre:

### I.2.1 Utilisation maximale des différents sens humains

Plus l'information sera représentée sous des formes différentes , plus elle aura de chances d'être perçue par tous les sens du mémorisant ( goût, odorat, ouïe, toucher, vue). Et comme, apparemment, une information est d'autant mieux mémorisée qu'elle est bien "ancrée" dans la mémoire, cela devrait favoriser une bonne mémorisation.

Le système devra donc présenter l'information pour qu'elle soit perceptible par les différents sens humains.

### I.2.2 Limite de la mémoire à court terme

Etant donné sa capacité restreinte, le système ne présentera pas plus de 7 nouvelles informations en une fois à l'élève.

### I.2.3 Temps de mémorisation

Il semble qu'il soit préférable de consacrer plusieurs courtes périodes à la mémorisation plutôt qu'une seule longue période (20 à 40 minutes de mémorisation pour 5 à 10 minutes d'arrêt). Le système veillera donc à ménager des pauses régulières pour l'étudiant.

### I.2.4 Rapidité de l'oubli

Puisque l'oubli est fonction, notamment, du temps, le système pourra fonctionner en "sachant" qu'une information risque d'être oubliée après une période plus ou moins longue.

## I.3 Conclusion

Les trois derniers points ne posent guère de problèmes et un système informatique peut facilement en tenir compte.

Le quatrième, l'utilisation des différents sens humains lors de la présentation des données à mémoriser, est plus difficile à concevoir. La conception d'un système qui pourrait présenter différentes odeurs ou agir sur le sens du toucher est trop complexe à envisager.C'est pourqoi, il faudra ici se fixer certaines limites (cfr paragraphe II.2).

# CHAPITRE II : ANALYSE D'UN COMPORTEMENT DE MEMORISATION.

## II.1 INTRODUCTION

Etant donné qu'il n'existe pas de théorie scientifique expliquant le fonctionnement de la mémoire, étant donné que nous ne connaissons pas la structure de ce que nous mémorisons et que nous ne connaissons pas, non plus, le processus de mémorisation, nous allons présenter, dans ce chapitre, deux scénarios fictifs. Nous espérons, grâce à ceux-ci, pouvoir mettre en évidence d'autres hypothèses, qui pourraient servir à la conception de notre système, et qui expliqueraient le fonctionnment de la mémoire.

Notons que nous avons volontairement choisi des exemples très simples et qu'aucun professeur ne perdrait son temps à un tel travail.

**Premier scénario.:** l'apprentissage de deux mots néerlandais.

Prof: Voiture se traduit en néerlandais par wagen.
Voiture d'occasion se traduit en néerlandais par tweedehandswagen

Prof: Mon papa roule avec une voiture d'occasion se traduit en néerlandais par : Mijn vader rijdt met een tweedehandswagen.

Prof: Comment dit-on voiture en néerlandais ?

Elève: Wagen.

Prof: C'est bien.

Prof: Comment dit-on voiture d'occasion en néerlandais ?

Elève: Je ne sais pas.

Prof: Comment dit on voiture ?

Elève: Wagen.

Prof: C'est bien

Prof: Voiture d'occasion se traduit en néerlandais par tweedehandswagen.

Prof: Dans la phrase "Mijn vader rijdt met een tweedehandswagen" que signifie tweedehandswagen ?

Elève: Voiture d'occasion.

Elève: Voiture d'occasion se traduit en néerlandais par tweedehandswagen ?

Prof: Oui.

Prof: As-tu compris ?

Elève: Oui.

**Second scénario:** la table de multiplication par 8.

Prof: 1 fois 8 égalent 8.

2 fois 8 égalent 16.

3 fois 8 égalent 24.

4 fois 8 égalent 32.

5 fois 8 égalent 40

Prof: Combien font 1 fois 8 ?

Elève: 8.

Prof: OK, et combien font 2 fois 8 ?

Elève: 16.

Prof: OK, et combien font 3 fois 8 ?

Elève: Je ne sais pas.

Prof: 3 fois 8 égalent 24.

Prof: Combien font 4 fois 8 ?

Elève: Je ne sais pas.

Prof: 4 fois 8 égalent 32.

Prof: Combien font 3 fois 8 ?

Elève: 32.

Prof: Non ! 3 fois 8 égalent 24.

Prof: Combien font 5 fois 8 ?

Elève: 40.

Prof: Combien font 3 fois 8 ?

Elève: 24.

Prof: OK.

**Analyse de ces deux scénarios**

Avant d'en commencer l'analyse, il nous semble intéressant de rappeler que notre système ne permet pas de mettre en évidence des raisonnements qui pourraient faciliter la mémorisation. Nous pensons, notamment, aux moyens mnémotechniques.

L'exemple du vocabulaire néerlandais permet d'illuster ce que nous voulons dire:

Nous ne tenons pas compte du fait que le mot wagen fait partie du mot tweedehandswagen. Pour le système, il s'agit de deux mots différents et indépendants.

Ce choix a été fait car il nous semble que le système est déjà suffisamment complexe comme cela.

Analysons maintenant le discours des deux interlocuteurs pour voir ce que nous pouvons en conclure.

1) Le professeur.

Il pose les questions suivantes :

- Comment dit-on voiture en néerlandais ?
- Comment dit-on voiture d'occasion en néerlandais ?
- Dans la phrase "Mijn vader rijdt met een tweedehandswagen" que signifie tweedehandswagen ?
- As-tu compris ?
- Combien font 1 fois 8 ?
- OK, et combien font 2 fois 8 ?
- OK, et combien font 3 fois 8 ?
- Combien font 4 fois 8 ?
- Combien font 3 fois 8 ?
- Combien font 5 fois 8 ?

Il donne les informations suivantes (y compris encouragement) :

- Voiture se traduit en néerlandais par wagen.
- Voiture d'occasion se traduit en néerlandais par tweedehanswagen
- Mijn vader rijdt met een tweedehandswagen se traduit en français par : Mon papa roule avec une voiture d'occasion.
- C'est bien.
- Oui.
- 1 fois 8 égalent 8.
- 2 fois 8 égalent 16.
- 3 fois 8 égalent 24.
- 4 fois 8 égalent 32.
- 5 fois 8 égalent 40
- Non ! 3 fois 8 égalent 24.
- OK.

2) L'élève

Il pose les questions suivantes :

- Voiture d'occasion se traduit en néerlandais par tweedehanswagen ?

Il donne les informations suivantes :

- Wagen
- Voiture d'occasion
- Oui
- Je ne sais pas .

Regardons à présent ce que le professeur utilise pour permettre à l'élève de mémoriser les deux mots de vocabulaire de néerlandais ou la table de multiplication par 8.

- Les mots de vocabulaire à faire mémoriser : wagen et tweedehanswagen.
- Les chiffres à faire mémoriser : 8, 16, 24, 32, 40.
- Il donne à l'étudiant la traduction des mots.
- Il donne la valeur des multiplications.
- Il pose des questions à l'élève, pour évaluer sa connaissance; suivant les réponses fournies, il pose d' autres questions ou il donne des informations.

Des scénarios ci-dessus, nous pouvons tirer les observations suivantes:

1 Il faut qu'il y ait une matière à faire mémoriser et notre système devra en avoir une représentation. Nous allons, comme nous l'avons déjà dit, permettre au professeur de décrire **synthétiquement** la connaissance à faire mémoriser.

2 Le professeur fictif tient compte de la manière dont l'élève étudie. Il sait si un fragment de connaissance est connu ou il tient compte d'une éventuelle erreur de l'élève.

Nous considérons que ces événements font partie de l'environnement de la mémorisation. Notre système devra aussi en tenir compte.

3 Le professeur applique des règles pour aider l'étudiant. Par exemple: "s'il ne répond pas bien, je lui reposerai telle question".

Il faudra que le système les connaisse également.

C'est le professeur qui les lui fournira. Il devra pouvoir le faire, comme pour la description des connaissances, de façon **synthétique.**

Notre système devra utiliser ces trois sources d'informations pour fonctionner. Nous allons maintenant les analyser plus en détail.

## II.2 Les connaissances à faire mémoriser.

Qu'avons-nous l'habitude d'étudier par coeur ?

Il y a, d'abord, les listes des codes journaliers : un numéro de compte en banque, une plaque d'immatriculation, un numéro de carte d'identité...

Il y a, ensuite, les données que l'on utilise régulièrement et que l'on mémorise pour ne pas devoir les chercher : les numéros de téléphone et adresses des proches...

On peut aussi mémoriser des listes qui facilitent le travail : une liste d' abréviations des produits que l'on vend, le morse pour les télégraphes...

Toutes ces connaissances ont un point commun : elles sont, chaque fois, associées à un objet ou à un individu.

Exemple

mon compte ---- 010 0045678 87

ma voiture ---- HER 456

téléphone voisin --- 65-74-53

produit beaujolais nouveau ---- BJ

On peut également mémoriser d'autres classes d'éléments plus complexes : la liste des synonymes d'un mot, les prénoms des enfants d'un ami...

Exemple

les Dupondt : pierre ,yves, suzanne et jean

voler : prendre, dérober

Mais on mémorise aussi des éléments aux structures encore plus complexes : les paroles d'une chanson ou une définition, par exemple.

Constatons, dans toutes ces connaissances que l'on mémorise, les deux points communs suivants: d'une part, pour mémoriser, il faut rattacher la nouvelle connaissance à une ancienne; d'autre part, on structure la connaissance. En effet même si on voulait faire apprendre par coeur un texte chinois à quelqu'un qui ne connaît pas cette langue, il verrait,pour l'étudier, une suite de petits dessins.

Nous pouvons imaginer facilement que la structure sera assez importante pour la mémorisation de tout un vocabulaire d'une langue utilisant des phrases complètes ou pour la mémorisation de schémas complexes.

## II.2.1 Représentation de la connaissance

Dans le cadre de ce système, un utilisateur-élève et un utilisateur-professeur devront manipuler des représentations de connaissance à mémoriser. L'un les décrira, l'autre les mémorisera.

Nous allons choisir dans cette partie un mode de représentation.

Au préalable, il nous semble intéressant de s'arrêter un instant pour se demander : "Que mémorise-t-on et quelle représentation en a-t-on ?"

| Exemple | **Représentation** | **Objet à mémoriser** |
|---|---|---|
| - | Une lettre | a |
| - | Un chiffre | 2 |
| - | Des lettres: un mot | papa |
| - | Des chiffres: un nombre | 014 |
| - | Des mots: un texte | Pierre mange. |
| - | Des symboles | # |
| - | Des images | un paysage |
| - | Des voix | un ami |
| - | Des odeurs | un fromage |
| - | Des goûts | l'ail |
| - | Des sensations | chaud et froid |

C'est-à-dire, tout ce qui se traduit par des sensations.

Il nous semble évident que l'avancement actuel de l'informatique, en général, et de la micro-informatique, en particulier, ne peut encore aider à une mémorisation utilisant chacun des sens humains, du moins à des coûts raisonnables.

### II.2.1.1 Quelles représentations ?

Nous considérons que l'outil dont on dispose est un simple ordinateur. Les outils périphériques sont dès lors réduits à un écran et une imprimante. L'utilisation d'un enregistreur pourrait aussi être envisagée. On ne peut alors garder que les représentations suivantes :

- Une lettre
- Un chiffre
- Des lettres: un mot
- Des chiffres: un nombre
- Des mots: un texte
- Des symboles
- Des dessins
- Des sons
- Des voix préenregistrées

Le tout utilisant une certaine dynamique.

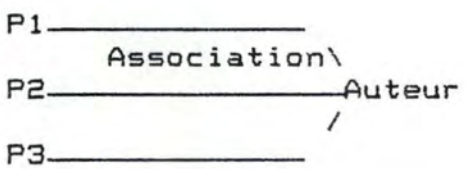
### II.2.1.2 Quelle structure?

Nous proposons d'établir ici une relation qui susceptible d'être celle utilisée par la mémoire : l'association de concepts mémorisés.

## L'association

Lorsqu'un concept est proposé à une personne, il y associe un autre concept.

exemple : un élève doit étudier trois titres de poésie (P1,P2,P3) d'un même auteur.

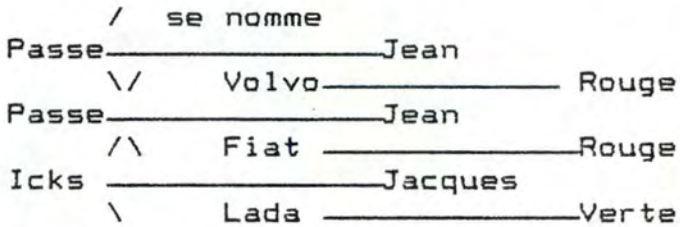



L'ensemble de la connaissance d'un individu peut ainsi être représenté, théoriquement, par une gigantesque structure. Chaque association peut être également un élément de la structure.

Si nous généralisons le schéma, nous pouvons dire que chaque élément est soit une extrémité d'un chemin, soit un chemin.

Un chemin peut donc être l'extrémité d'un autre chemin.

exemple : Mémoriser que Mr Jean Passe a deux voitures, une Fiat rouge et une Volvo rouge et que Mr Jacques Icks possède une Lada verte.



nous avons les extrémités :

Volvo,Fiat,Lada,Rouge,Verte,Jean,

Jacques,Passe,Icks.

nous avons les chemins :

Jean,Jacques,se nomme

Remarquons que ce schéma n'est pas "le"schéma. D'autres représentations de la connaissance à mémoriser étaient possibles.

### II.2.1.3 Présentations possibles de la connaissance.

Il faudrait idéalement pouvoir interroger ou informer l'élève sur des associations existant entre des concepts représentés par des lettre, chiffre, mot, nombre, texte, symbole, image, son et voix. Chacune des représentations visent une sensation particulière.

Nous devons rester les pieds sur terre et nous rendre compte qu' utiliser toutes ces présentations est utopique. Voyons ce que nous garderons.

Etant donné le caractère semblable de certaines représentations, nous proposons d'établir ici un regroupement sous trois termes: le texte, le dessin et la voix.

**Un texte** est une suite de caractères (lettres, chiffres, ponctuation).

**Un dessin** est une image représentée. Un symbole peut être considéré comme un dessin.

**La voix** est une suite de sons intelligibles.

On peut maintenant définir les trois modes de représentation que notre système pourrait théoriquement utiliser.

1 est expliqué par

2 est dessiné par

3 est dit par

## II.2.2 La mémorisation.

Dans cette partie, nous définissons ce qu'est mémoriser.

### II.2.2.1 Connaître.

Connaître une structure, c'est pouvoir la redonner de mémoire.

Exemple : mémoriser le mot anglais "APPLE"

"APPLE" —(est dit par)→ "POMME"

"APPLE" —(est dessiné par)→ dessin d'une pomme

On représentera tantôt le mot "pomme", tantôt le dessin d'une pomme. Si, dans un nombre de cas satisfaisant, l'élève répond par le mot "apple", nous considérerons qu'il a mémorisé le mot.

Plus l'élève aura pu créer des liens entre une nouvelle notion et sa structure cognitive, mieux cette notion sera mémorisée.

Notre système devra essayer de faire retenir tous ces liens à l'étudiant. Cela paraît cependant compliqué vu la diversité des individus: un lien significatif pour une personne ne l'est pas nécéssairement pour une autre. C'est le professeur qui pourra éventuellement tenir compte de ce fait dans la description des règles qu'il donnera au système (voir "Métaconnaissance").

### II.2.2.2 Parcours d'une structure et historique.

En apprenant une structure, l'élève risque de privilégier certaines associations. Nous considérerons que celles qui reviendront le plus souvent seront les plus connues. Pour que la mémorisation soit la plus efficace possible, il faut aider l'élève à mémoriser celles qu'il connaît le moins.

L'historique d'une association, c'est le nombre de fois qu' elle a été choisie pour retrouver l'extrémité de l'arc à memoriser.

Exemple : Soit un éleve qui, face au dessin d'une pomme, répond "apple" (A1) et face au mot "pomme" reste muet(A2).

Nous considérons que le parcours A1 est établi et donc mémorisé. A2 ne l'étant pas encore.

### II.2.3 Description synthétique de la connaissance

Lorsque le système pourra, de lui-même, trouver les informations à faire mémoriser, il ne sera pas nécessaire au professeur de lui donner exhaustivement la connaissance.

Néanmoins, l'objectif reste d'aider à mémoriser des relations et le système ne donnera donc pas ses méthodes pour trouver les concepts à mémoriser.

Exemple Mémoriser le b a ba du calcul mental sans devoir calculer les tables de multiplication.

Dans de tels cas, le professeur pourra décrire la matière synthétiquement. C'est le cas de la maquette que nous développerons.

### II.2.4 Conclusion

Etant donné le caractère relationnel de la structure de représentation proposée ci-dessus et le fait qu'elle peut être importante, nous suggérerons l'emploi d'une base de données relationnelle. Cet emploi semble adéquat pour représenter les connaissances à mémoriser. Nous rappelons qu'il s'agit ici de représenter la structure de connaissance qui sera en mémoire de l'ordinateur. Ce n'est pas pour autant ce qu'il y aura dans la mémoire de l'élève.

Toutefois, il faudra ajouter une certaine dynamique au système.

En effet, le système devra connaître les points d'accès de la structure originale pour y rattacher la nouvelle connaissance.

Il faudra donc lui dire les éléments (items et relations) de la structure qui sont déjà dans la mémoire de l'élève.

Ainsi, dans une relation entre trois items, plusieurs accès sont possibles:

rel(item1,item2,item3)

accès 1 : à partir de item1
accès 2 : à partir de item2
accès 3 : à partir de item3

ou toute combinaison à partir des 2 parties.

accès 4 : à partir de item1 et item2
accès 5 : à partir de item1 et item3
accès 6 : à partir de item2 et item3

## II.3 La métaconnaissance.

Comme nous l'avons vu, le professeur utilise certaines règles pour aider l'étudiant lors de la mémorisation. En fonction d'événements divers, il modifie son comportement en donnant telle information ou en posant telle question à l'élève.

Toutes les règles utilisées comprennent deux parties: la ou les conditions d'application et l'attitude à adopter.

L'ensemble de ces règles forme ce que nous appelons la métaconnaissance, à savoir la connaissance au sujet de la connaissance.

### II.3.1 Role de la métaconnaissance.

La métaconnaissance constitue, en fait, l'élément essentiel du système aidant à la mémorisation. Elle sert à décrire le comportement du système par rapport à l'élève. C'est le professeur qui, de façon déclarative (et non pas procédurale), fournit les informations nécessaires.

### II.3.2 Composition.

Le professeur devra fournir 5 notions.

- La définition des questions que l'élève pourra poser au système.
- La définition des questions que le système pourra poser à l'élève (c'est-à-dire une certaine représentation des questions) ainsi que le pourcentage de réussite associé.

- La définition des informations que le système pourra donner à l'élève (y compris les encouragements).
- La définition des informations que l'élève pourra donner au système (ex : je connais déjà ceci).
- Les conditions dans lesquelles questions et informations peuvent apparaître: quand le système peut-il poser telle question à l'élève et quand l'élève peut-il poser telle question au système (idem pour les informations).

Des précisions quant à ces conditions seront apportées plus loin.

Il est clair que, si le professeur n'autorise aucune question à l'élève, le système sera totalement maître du parcours de la mémorisation (système directif). Inversément, l'élève pourrait être le maître si le système ne pouvait lui poser aucune question.

Ces deux cas extrêmes semblent cependant peu réalistes. En effet, comment envisager un enseignement dans lequel soit le professeur, soit l'élève, ne pourrait poser de questions à l'autre?

Remarquons que le professeur peut donc offrir la possibilité à l'élève de diriger le système. Nous refusons donc toute directivité totale de l'ordinateur.

A partir des cinq notions vues plus haut, le système pourra tirer des conclusions qui lui permettront de fonctionner.

II.3.2.1 La réponse aux questions.

Le système pourra trouver, à partir de la connaissance qu'il a, les réponses adéquates aux questions qu'il posera à l'élève ou à celles que l'élève lui posera.

II.3.2.2 Dynamique de mémorisation.

Grâce aux questions permises et à leurs conditions d'application, le professeur pourra donner des indications sur le déroulement de la mémorisation, et dès-lors une maîtrise pédagogique du déroulement de la mémorisation sera possible. C'est ainsi que le dynamisme de la mémorisation sera introduit.

L'enseignant pourrait, par exemple, donner au système le nom de relations qui devraient être mémorisées avant d'autres et ainsi en privilégier certaines.

II.3.2.3 Plusieurs métaconnaissances pour une même connaissance.

Grâce aux questions permises et aux conditions d'application des questions et informations définies dans la métaconnaissance, le système peut connaître la partie correspondante de la structure que le professeur souhaite faire mémoriser par l'élève. Celle-ci pourrait être différente de la structure de connaissance mise en mémoire du système par le professeur (si certaines relations ne se retrouvent pas dans les questions à

poser et informations à donner). Cela permettrait au système d'être plus rapide que s'il devait manipuler une base de données comprenant beaucoup plus d'informations que nécessaire au didacticiel utilisé.

Autre intérêt: on pourrait facilement envisager plusieurs didacticiels qui travailleraient sur une même structure de connaissance. Rappelons, en effet, que la partie de la structure que le système aidera à mémoriser est directement fonction des relations utilisées dans la métaconnaissance.

II.3.2.4 L'évaluation.

III.3.2.4.1 L'évaluation du système.

Grâce aux réponses qu'il aura reçues à ses questions, le système pourra évaluer la connaissance de l'élève. On supposera que la connaissance sous-jacente à une question a été mémorisée par l'élève s'il a pu répondre à un certain nombre de questions se rapportant à cette connaissance et si les réponses fournies étaient statistiquement satisfaisantes, une bonne réponse ne suffisant pas pour que la connaissance soit supposée mémorisée.

La structure à mémoriser sera supposée connue si, à toutes les questions posées (Qi), l'élève a atteint une cote plus grande ou égale au seuil (Si) défini par le professeur.

**Syst connu <=> Qi>= Si**

Les questions posées à l'élève seront celles définies par le professeur dans la métaconnaissance dans laquelle le ou les paramètres auront été instanciés (puisqu' elles auront été définies synthétiquement; cfr III.3.3) par le système. Elles porteront donc sur les occurrences de relation et à chaque question sur une occurrence de relation correspondra donc une cote (évaluation de la connaissance d'une question).

L'évaluation sera satisfaisante si le seuil (S) de connaissance défini par le professeur pour cette question est atteint, c'est-à-dire si :

**éval Q <=> (ancéval Q +(-4,4))*μ*N2/N1 >= S**

Explicitons cette formule d'évaluation.

Pour juger que l'élève à mémorisé la réponse à une question, il faut qu'il puisse fournir une bonne réponse à plusieurs reprises.

Il faut également tenir compte du moment où la dernière bonne réponse a été fournie : plus ce moment est éloigné, plus le risque que la réponse ait éte oubliée est grand. Voilà pourquoi tenir compte du temps dans l'évaluation de la connaissance est indispensable.

Pour attribuer la cote d'évaluation, nous procédons de la façon suivante :

Comme base, nous reprenons la dernière évaluation (ancéval), à laquelle nous ajoutons 4 si la réponse correcte à été donnée (voir ci-dessous). Sinon, nous retranchons 4.

Pour atteindre une cote de 10, il faudra donc que l'élève prouve trois fois, au moins, qu'il connaît la réponse à la question posée.

Pour tenir compte du temps, nous pondérons ensuite le tout par $\mu$. Plus l'évaluation ancéval sera ancienne, plus $\mu$ sera faible, ce qui permet de tenir compte du fait que "avec le temps, on oublie tout".

Enfin, nous multiplions le nombre obtenu par N1/N2, c'est-à-dire le nombre de bonne réponse obtenues à des questions portant sur cette occurence de relation (N1) divisé par le nombre de questions posées portant sur cette occurence (N2).

Ce dernier produit nous donne la nouvelle cote d'évaluation.

Remarquons une faiblesse à cette formule. Si l'étudiant s'est trompé dans une réponse et si la dernière évaluation de cette question est trop ancienne alors la nouvelle évaluation sera fort diminuée. Pourtant, il suffirait d'une simple réactivation pour lui permettre de se remémorer la notion déjà apprise.

| Temps écoulé depuis la dernière question portant sur l'occurence | Coefficient MU |
|---|---|
| plus de 365 jours | 0.2 |
| entre 32 et 365 jours | 0.5 |
| entre 1 et 31 jours | 0.7 |
| plus de 20 min, moins d'1 jour | 0.85 |
| entre 11 et 20 min. | 0.9 |
| entre 6 et 10 min. | 0.97 |
| moins de 6 minutes | 0.99 |

Nous avons dit que nous ajoutons 4 à la cote si la réponse fournie par l'élève était correcte. Mais ...

Qu'est ce qu'une bonne réponse ?

Cette question apparemment anodine cache en fait un grand problème que nous ne pourrons traîter suffisamment, faute de temps. En effet, quand peut-on dire qu'une connaissance à été mémorisée.
Voici ce que nous entendrons quand nous dirons qu'une réponse est correcte.

Le professeur, en posant une question, pourra s'attendre à trois types de bonnes réponses de la part de l'élève :

- toutes les réponses possibles et satisfaisantes à la question.

- des réponses possibles et satisfaisantes à la question

- une réponse en terme de oui ou de non.

Dans le premier cas, on supposera que la réponse de l'étudiant est correcte s'il a pu redonner toutes les réponses possibles.

Dans le deuxième cas, on supposera que la réponse de l'étudiant est correcte s'il a pu redonner au moins deux des réponses possibles et correctes.

Dans le dernier cas, on supposera que la réponse est correcte si l'étudiant a repondu correctement oui ou non.

II.3.2.4.2 Autoévaluation

L'élève pourra aussi s'autoévaluer en posant des questions au système ou en lui donnant des informations. Il verra ainsi si les réactions du système sont celles auxquelles il s'attend.

II.3.3 Description synthétique de la métaconnaissance.

Les questions, informations et conditions devront porter sur des relations et non sur des occurencs de relations. Le système décidera lors de la mémorisation des occurences à faire apprendre.

C'est pourquoi, nous parlons de description synthétique, les règles ne portant pas directement sur des fragments de relation.

II.3.4 Conclusion

Toutes ces questions et informations de la métaconnaissance vont permettre d'établir le dialogue entre l'élève et le système.

## II.4 L'environnement.

### II.4.1 Objectif.

Le système aura besoin de pouvoir suivre l'évolution de la mémorisation, pour pouvoir réagir comme il le doit. C'est pourquoi, il est impératif de savoir où l'étudiant se situe dans sa démarche de mémorisation. Nous appellerons environnement les notions utilisées autour de cet apprentissage. C'est ici que le sytème viendra voir le chemin parcouru par l'élève, à savoir l'évolution de la mémorisation, afin de pouvoir en tenir compte dans le choix de ses prochaines questions et informations.

### II.4.2 Composition.

L'environnement sera représenté par une structure qui reprendra un historique des quelques dernières questions posées à l'élève et de leurs réponses, ainsi qu' une base de données qui contiendra la matière et ce que l'étudiant en connaît déjà. Chaque occurrence de relation sera évaluée en cours de mémorisation (cfr paragraphe sur l'évaluation).

A chaque occurrence de relation sera associé un compteur donnant le taux de réussite de l'élève. Selon qu'une bonne ou une mauvaise réponse sera apportée à une question du système, il augmentera ou diminuera.

## II.5 Le moteur

Celui-ci devra gérer les informations dont il dispose. C'est lui qui choisira le chemin de la mémorisation avec comme objectif que la connaissance soit connue par rapport à l'évaluation définie ci-dessus (cfr paragraphe sur l'évaluation).

## II.6 Les outils de manipulation de la connaissance à mémoriser.

Le professeur ne sait pas ce qu'est une base de données relationnelle d'un point de vue informatique. Il faut donc l'aider, par des outils adéquats, à entrer les données à mémoriser sous cette forme. Toutefois, il doit rester conscient du caractère structuré de ce qu'il veut faire mémoriser.

A chaque connaissance à mémoriser correspond une structure; et le professeur devra la communiquer à l'ordinateur.

L'élève ne sait pas non plus ce qu'est une base de données relationnelle et il n'a pas à le savoir. C'est pourquoi il faudra lui fournir des outils lui présentant les données à mémoriser et des outils l'interrogeant sur l'état de ses connaissances.

## II.6.1 OUTIL D'AIDE A L' INTRODUCTION DES CONNAISSANCES A MEMORISER PAR UNE PERSONNE. INTERFACE PROFESSEUR-MACHINE

Il faudra demander au professeur les connaissances à mémoriser ainsi que la métaconnaissance. Différents langages devront être prévus afin de faciliter le dialogue entre l'enseignant et le système. La connaissance et les règles pourront être décrites de façon syntétique.

Il faudra que cette partie du programme soit la moins lourde possible pour le professeur. Puisque la solution proposée ici n'est qu'indicative, d'autres versions de cet interface devront être développées dans le but d'être plus conviviales.

### II.6.1.1 Les connaissances.

Le professeur doit pouvoir définir le nom des relations, la liste des items liés et leurs occurrences(le professeur devra donc avoir clairement structuré la connaissance pour pouvoir la donner au programme). Il faut que, dans les relations, certains concepts soient connus de l'élève mémorisant, afin de constituer une charnière entre ce que l'élève connaît déjà et la nouvelle matière (cfr réseau semantique).

ex: 1. la relation "est traduit par" entre deux mots sera donnée au système par :

est traduit par(string string)

2. Vader est traduit par papa

Moeder est traduit par maman

Voici une solution que nous proposons.

Dans une première colonne, le professeur donne le nom d'une relation. Dans les colonnes de droite, il donne les différentes occurrences de chaque item. Il y a autant de colonnes qu'il y a d'items pour la relation en cours.

Exemple de la table de multiplication.

| | | | |
|---|---|---|---|
| fois | 1 | 8 | 8 |
| | 2 | 8 | 16 |
| | 3 | 8 | 24 |
| | 4 | 8 | 32 |
| | 5 | 8 | 40 |

remarque : Pour ce genre de matière, la description sera facilitée car l'ordinateur peut calculer. Pour l'élève, il s'agira toujours de mémorisation par association et non par calcul.

Exemple du vocabulaire néerlandais.

| | | |
|---|---|---|
| Se traduit en néerl. | Voiture | Wagen |
| | Voiture d'occasion | Tweedehandswagen |

II.6.1.2 La métaconnaissance

Le professeur devra préciser les données relatives à la métaconnaissance telles que vues plus haut. Pour mémoire, il s'agit des questions permises au système et à l'élève, des informations permises au système et à l'élève ainsi que de leurs conditions d'application.

Notre but est de ne pas devoir demander au professeur qu'il entre une à une toutes les questions et informations possibles et imaginables. Il faudra donc que l'outil dispose d'un langage de description synthétique des questions et informations manipulées par le système.

Nous rappelons que c'est précisément en cela que se distingue notre système de ceux de l'EAO habituel, dans lequel toutes les questions doivent être prévues et décrites exhaustivement.

Langage de description synthétique des questions informations et conditions à leur application

Pour décrire ce langage nous procéderons de la façon suivante: nous verrons en premier lieu la sémantique, c'est-à-dire quel est l'objet manipulé et quelles sont les actions possibles sur cet objet, nous envisagerons, ensuite, une syntaxe possible à utiliser.

**II.6.1.2.1 Sémantique**

II.6.1.2.1.1: Objet manipulé: la règle.

Il y aura quatre types de règles.

- Les questions à poser
- Les informations à donner
- Les questions à accepter
- Les informations à accepter

I Les questions à poser à l'élève.

Elles ont pour but de décrire au système les questions qu'il pourra poser. Dans ce cas, le professeur devra dire:

- un nom de relation et s'il peut être demandé à l'élève par le système.
- pour chaque item, s'il est à connaitre ou s'il est supposé avoir une sémantique connue pour l'élève.
- les conditions d'application.

## II Les informations à donner à l'élève.

Leur but est de décrire au système les informations que celui-ci pourra donner. Nous en distinguerons deux types: les informations relatives à la connaissance à faire mémoriser (A) et celles relatives à une certaine pédagogie (B).

A.. relativement à la connaissance.

Le professeur devra dire :

- un nom de relation et s'il peut être demandé à l'élève par le système.
- les conditions d'application.

B.. relativement à la pédagogie :

Le professeur devra donner :

- une phrase, suite de caractères et de blancs
- les conditions d'application.

## III Les questions à accepter de l'élève.

Elles ont pour but de décrire au système les questions qu'il pourra accepter de l'élève. Dans ce cas, le professeur devra dire:

- un nom de relation et s'il peut être demandé au système par l'élève.
- pour chaque item, s'il peut être demandé par rapport aux autres.
- les conditions d'acceptation de la part du système.

## IV Les informations à accepter de l'élève.

Leur but est de décrire au système les informations qu'il pourra accepter de l'élève. Nous en distinguerons deux types: celles relatives à la connaissance à faire mémoriser (A) et les réponses aux questions du système (B).

A..informations relatives à la connaissance.

Le professeur devra donner :

- un nom de relation et ses items.

B.. réponses aux questions du système :

Celles-ci ne sont pas à définir par le professeur puisque le système les connaît déjà.

### II.6.1.2.1.2: Les conditions d'application et d'acceptation

On va ici essayer de recréer les conditions qu'utilise un professeur avant d'interroger un élève. Voici, en quelques exemples, des conditions régulièrement utilisées.

ex1: S'il connaît telle chose alors je pose telle question

ex2: Si, à telle question, l'élève répond incorrectement, alors je lui pose telle autre question.

ex3: Si j'ai telle condition, alors je ne lui pose pas telle question.

ex4: S' il étudie telle matière, alors je refuse telle question de l'élève .

ex5: S'il étudie telle matière, alors j'accepte telle question de l'élève. ...

Les conditions permettront donc de faire référence à la valeur de la dernière question (cfr ex2).

Il faut aussi introduire la notion de chronologie en cours de mémorisation dans la définition des connaissances. En effet, une même réponse à une même question entraîne un comportement différent du professeur selon le moment où elle se produit.

Voici, en termes plus formalistes, les différents cas de figure que nous avons relevés:

Les conditions simples.

Q = Question  I = Information

i,j,k = identification de Q et I

e = éleve  s = système  c = concept conu

| | |
|---|---|
| Qi,s + Ii,e,c => Qj,s | Qi,s + Ii,e,c => Ij,s |
| Qi,s + not Ii,e,c => Qk,s | Qi,s + not Ii,e,c => Is,k |
| Qi,e,c + Ii,s => Qj,s | Qi,e,c + Ii,s => Ij,s |
| Ii,e,c => Qj,s | Ii,e,c => Ij,s |
| Ii,e,c => not Qj,s | Ii,e,c => not Ij,s |
| Ii,s => Qj,s | |
| Ii,s => not Qj,s | |

Traduction : par ex, Qi,s + Ii,e,c => Qj,c

Si, à la question i du système, l'élève a apporté une réponse i et s' il connaît le concept c, alors il faut lui poser la question j.

Toute combinaison logique de conditions simples pourra aussi être introduite comme condition d'application.

#### II.6.1.2.2 Syntaxe.

La syntaxe utilisée devra être la plus simple possible pour le professeur. Actuellement, nous proposons d'utiliser la syntaxe définissant les clauses du langage Prolog, car elle permet d'inclure des conditions avant l'action à accomplir. Le moteur prolog faisant automatiquement échouer une règle si une des conditions n'est pas réalisée.

## II.6.2 OUTIL D'AIDE A LA PRESENTATION DES DONNEES A MEMORISER ET DE L'ETUDE. INTERFACE ELEVE-MACHINE

. Il faudra prévoir un outil pour présenter, à l'élève, les connaissances à mémoriser.

ex : dans la phrase: "Mijn vader rijdt met een tweedehandswagen", le mot "tweedehandswagen" se traduit par "voiture d'occasion".

Un outil idéal pourrait toutefois être beaucoup plus sophistiqué en prévoyant la présentation de dessins, textes et voix.

. Il faudra ici aussi prévoir un langage permettant le dialogue entre l'élève et le système.

# CHAPITRE III : SPECIFICATION DU SYSTEME

## III.1 INTRODUCTION

Le programme a pour objectif d'apporter une aide efficace à des étudiants qui souhaitent mémoriser une matière (voir ci-dessous).

Il comprendra donc deux applications distinctes: la première qui permettra au professeur de donner à l'ordinateur la matière qu'il faudra faire mémoriser par les élèves; la seconde qui sera celle de la mémorisation. Le lien entre les deux étant bien évidemment la matière à faire mémoriser.

Analysons tout d'abord celle-ci.

## III.2 La matière

Sous l'expression "matière à faire mémoriser" nous avons remarqué deux notions fondamentales: la connaissance à faire mémoriser (III.2.1) et la manière dont l'ordinateur pourra aider l'étudiant dans sa phase d'apprentissage (III.2.2).

### III.2.1 La matière à faire mémoriser.

Celle-ci sera représentée par un schéma relationnel. En effet, nous avons, en effet, constaté (cfr chapitre 2) que ce mode de représentation paraissait très adéquat, étant donné le caractère relationnel de la mémorisation.

Un interface professeur-ordinateur permettra au premier d'entrer sa matière sans trop de peine sous cette forme.

Ce sera au professeur d'estimer si, dans le contexte de sa matière, ce mode de représentation est ad hoc.

Il pourra la communiquer d'une manière synthétique si la matière le permet.

### III.2.2 La métaconnaissance.

Nous entendons par métaconnaissance toutes les informations qui permettront a l'ordinateur de savoir comment guider l'étudiant.

Le professeur devra donner les indications suivantes: les questions et informations que l'ordinateur pourra donner ou accepter de l'élève, les conditions pour que celles-ci soient acceptables et les conditions à satisfaire pour que le système considère que l'élève a bien mémorisé sa matière.

Il pourra communiquer ces questions, informations et conditions de façon synthétique en utilisant les relations et non leurs occurences.

## III.3 Mémoriser une matière

Il convient de définir ce que nous entendrons par ce verbe. Avoir mémorisé une matière sera pour nous être capable de redonner précisément une ou plusieurs représentations d'un objet que l'on est censé connaître.

Il s'agit donc d'effectuer un simple travail d' association entre représentations d'objet. Le système s'attachera à aider l'élève uniquement pour ce genre de mémorisation.

## III.4 Objectif du système

Le système se voudra être d'une aide efficace pour l'étudiant qui doit mémoriser la matière définie par le professeur. Pour se faire, il guidera l'étudiant en utilisant la métaconnaissance.

. Il ne permettra pas de mettre l'accent sur des "ficelles" de mémorisation telles que les moyens mnémotechniques, mais présentera les associations entre les concepts à mémoriser.

. Il appliquera les deux points suivants (cfr chap.1):

1) ne pas donner à l'étudiant plus de sept nouvelles informations en une fois.

2) ménager une pose à l'étudiant toutes les 40 minutes, celui-ci pouvant toutefois s'arrêter quand il le souhaite.

Pour réaliser cela, il disposera d'un outil pouvant lui donner les 5 derniéres questions posées à l'élève ainsi que les réponses apportées (historique).

Dans certaines conditions (définies par le professeur), il sera capable d'accepter des questions de l'élève et d'y répondre.

Il pourra également évaluer la connaissance de l'élève et ainsi sélectionner les parties non encore mémorisées ou celles qui ont de fortes chances d'avoir été oubliées.

## III.5 Evaluation de la connaissance de l'élève.

Un système d'évaluation sera mis au point. Pour chaque relation de la connaissance, le professeur aura défini le seuil à atteindre (entre 0 et 10) pour considérer que l'étudiant a mémorisé . La formule de calcul de la cote est celle du paragraphe 3 du chapitre II.

Celle-ci sera évaluée en fonction des réponses apportées aux différentes questions portant sur la connaissance: la qualité, le nombre et la répartition chronologique étant comptabilisés.

## III.6 L'interface professeur-système.

Celui-ci est à définir selon l'implémentation. Il peut, par exemple, être conforme à la proposition faite dans le paragraphe 6.1 du chapitre II.

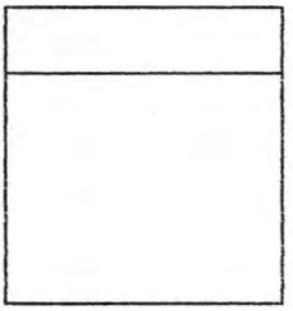
## III.7 Shéma global entité-association du système

Pour définir rigoureusement le sens des données du système, nous avons choisi d'utiliser un modèle de structuration des informations : le modèle entité association.

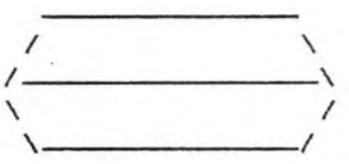
En effet, celui-ci rencontre aujourd'hui les suffrages de beaucoup de concepteurs de systèmes informatiques.

Basé sur les concepts d'**entité**, d'**association** et d'**attribut**, il a pour vocation de modéliser les informations du réel.

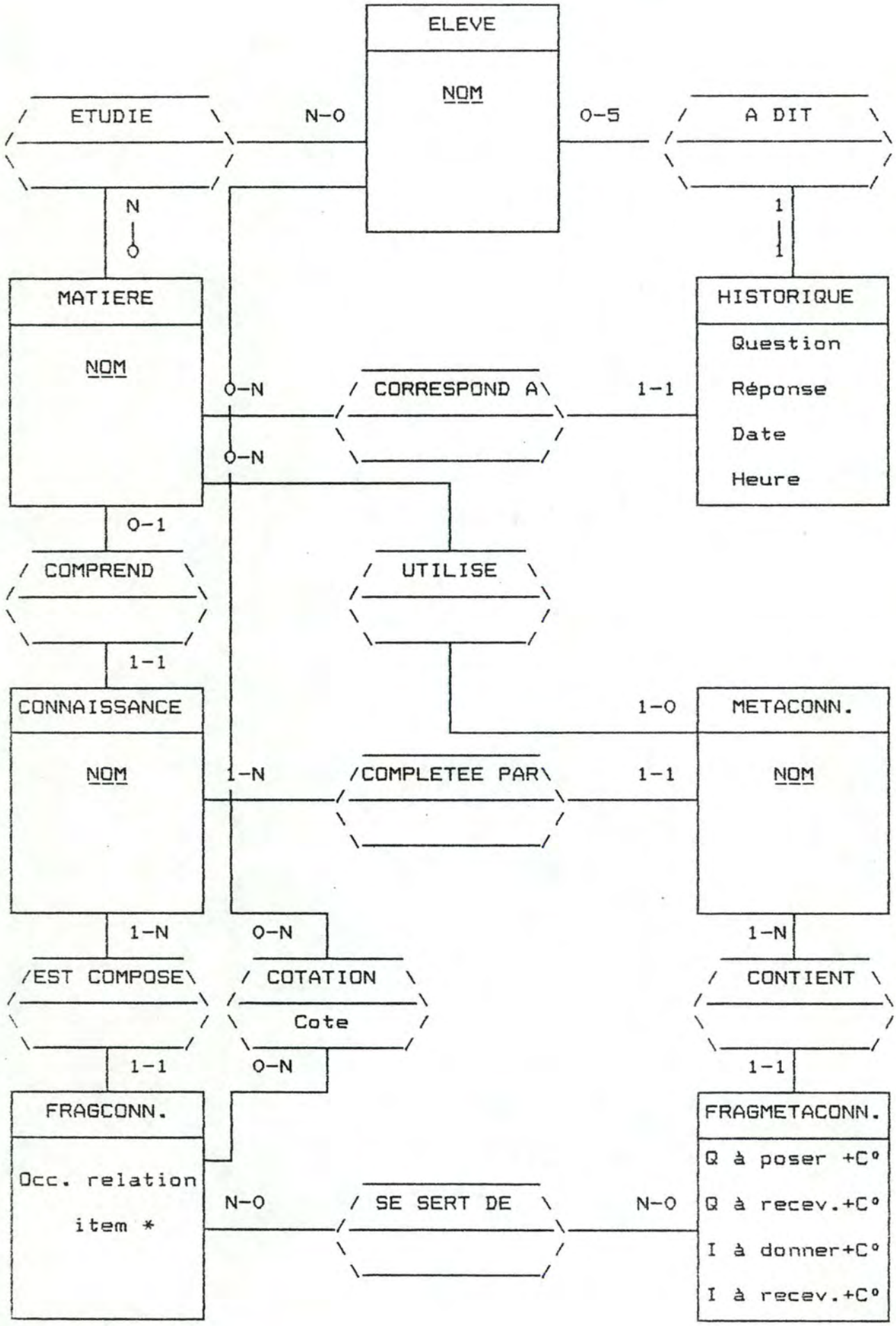
représentation d'un
type d'entité



représentation d'un
type d'association



Pour plus de détails concernant ce modèle, nous proposons au lecteur de se référer au livre mentionné en 2 ou à tout autre ouvrage présentant le modèle E/A.


ELEVE
NOM
ETUDIE
N-O
O-5
A DIT
N
1
1
MATIERE
NOM
HISTORIQUE
Question
Réponse
Date
Heure
O-N
CORRESPOND A
1-1
O-N
O-1
COMPREND
UTILISE
1-1
CONNAISSANCE
NOM
1-O
METACONN.
NOM
1-N
COMPLETEE PAR
1-1
1-N
O-N
1-N
EST COMPOSE
COTATION
Cote
CONTIENT
1-1
O-N
1-1
FRAGCONN.
Occ. relation
item *
FRAGMETACONN.
Q à poser +C°
Q à recev.+C°
I à donner+C°
I à recev.+C°
N-O
SE SERT DE
N-O

## III.8 Définition des types d'entité et type d' association.

### III.8.1 Les types d'entités

**ELEVE:** Toute personne utilisant le système d'aide à la mémorisation pour mémoriser une matière.

**MATIERE:** Une représentation des connaissances à faire mémoriser et des métaconnaissances correspondantes

**HISTORIQUE:** Ensemble des 5 dernières questions et réponses apportées par un élève.

**CONNAISSANCE:** Ensemble de la connaissance à faire mémoriser par des élèves.

**METACONN.:** Ensemble des notions utiles au système pour qu'il puisse aider l'élève dans sa tâche de mémorisation des connaissances.

**FRAGCONN.:** Une notion à faire mémoriser par les élèves.

**FRAGMETACONN.:** Ensemble de règles utiles au système pour aider l'élève à mémoriser un fragment de connaissance.

Rem: Nous parlons parfois de règle au lieu de fragment de métaconnaissance.

### III.8.2 Les types d' associations.

**ETUDIE**(Elève,Matière) :

Tel élève étudie telle matière.

**A DIT**(Elève,Historique) :

A tel élève correspond tel historique.

**CORRESPOND A**(Matière,Historique) :

Tel historique correspond à telle matière.

**COMPREND**(Matière,Connaissance) :

Telle connaissance fait partie de telle matière.

**UTILISE**(Matière,Métaconn.) :

Telle matière s'apprend en utilsant telle métaconnaissance.

**COMPLETEE PAR**(Connaissance,Métaconn.) :

Telle métaconnaissance complète telle matière.

**EST COMPOSE DE**(Connaissance,Fragconn.) :

Tel fragment de connaissance fait partie de telle connaissance.

**COTATION**(Elève,Fragconn.) :

Tel élève obtient telle cote pour la mémorisation de tel fragment de connaissance.

**SE SERT DE**(Fragconn.,Fragmétaconn.) :

Tel fragment de métaconnaissance sert à aider la mémorisation de tel fragment de connaissance.

**CONTIENT**(Métaconn.,Fragmétaconn) :

Tel fragment de métaconnaissance appartient à telle métaconnaissance.

## III.9 Types des attributs

Nom d'élève : - Nom : 15 caractères
- Prénom : 15 caractères

Nom de matière : 10 caractères

Question : texte

Information : texte

Relation : 20 caractères

Item : 20 caractères

cote : réel

Nom de connaissance : 10 caractères

Nom de métaconnaissance : 10 caractères

Date : entier de forme JJMMAA

Heure: entier de forme HHMMSS

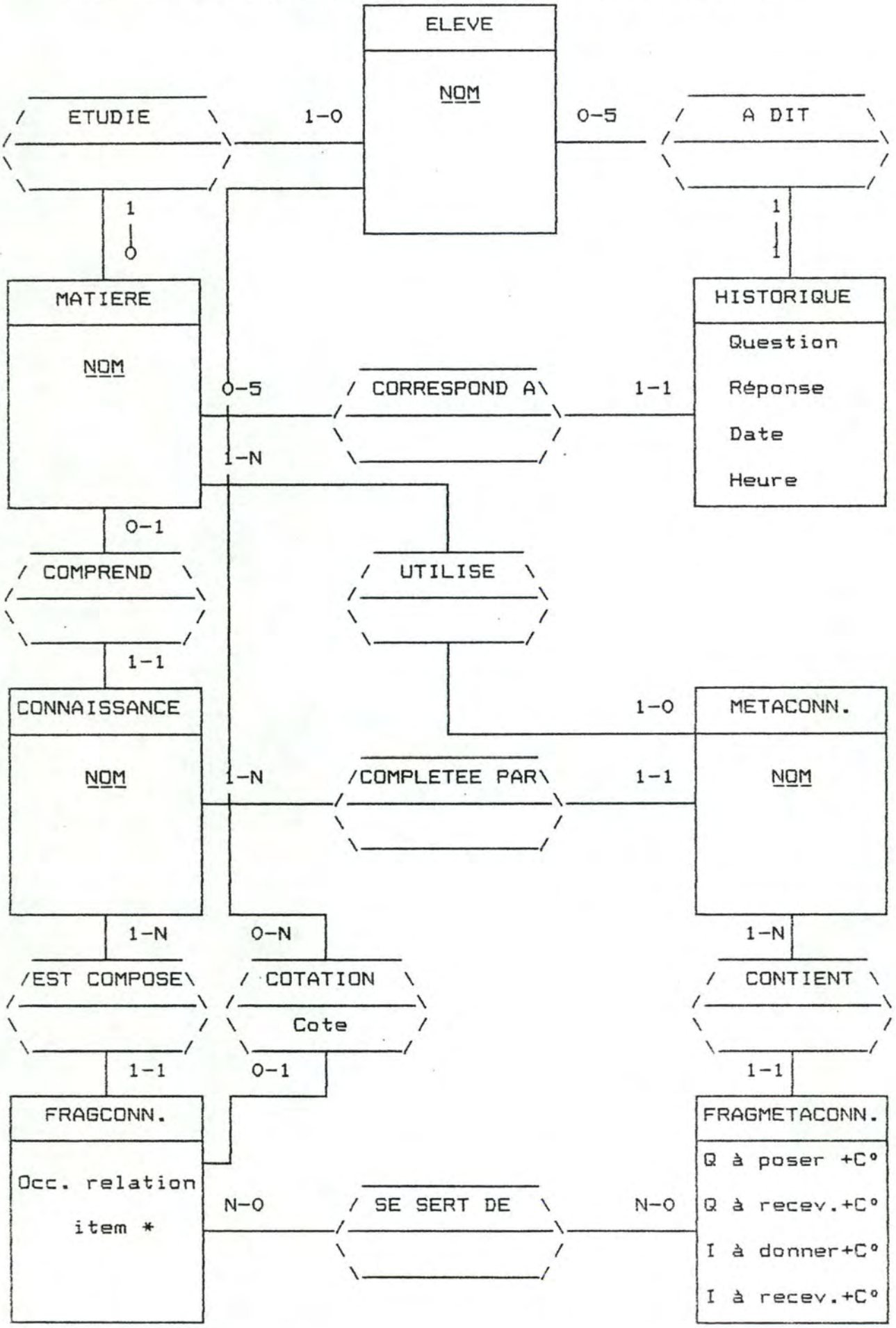
## III.10 Restriction à la spécification fonctionnelle

Etant donné que les écoles ne disposent aujourd'hui que d'un matériel très peu sophistiqué, il faut se rendre compte que le développement d'un tel système est impossible pour celle-ci.

C'est pourquoi, nous proposons ici une restriction au système en ajoutant ce qui suit à la spécification du système : le programme permettra à **un** étudiant de mémoriser **une et une seule** matière définie par le professeur.

Cela ne nuira pas à la réalisation du système, mais il faudra prévoir une copie par étudiant.

## III.11 Shéma entité-association restreint du système



## CHAPITRE IV : ARCHITECTURE LOGICIELLE.

Nous avons choisi de développer ce système sous forme d'une architecture modulaire.

En effet, cela offre de nombreux avantages :

- une forte opacité permettant de cacher l'information dans un module.
- un faible degré de couplage entre les différents modules.
- une forte cohésion (à savoir, une interdépendance la plus grande possible entre condition et action du module).

La relation employée entre modules est la relation **utilise** définie par :

Soit A, B : 2 modules.

A utilise B <=> A est nettement plus simple, d'un point de vue des spécifications, de la conception et de la maintenance s'il peut utiliser B.

ET B n'est pas beaucoup plus compliqué parce qu'elle ne peut utiliser A.

ET il n'existe pas de sous-système utile incluant B et non A.

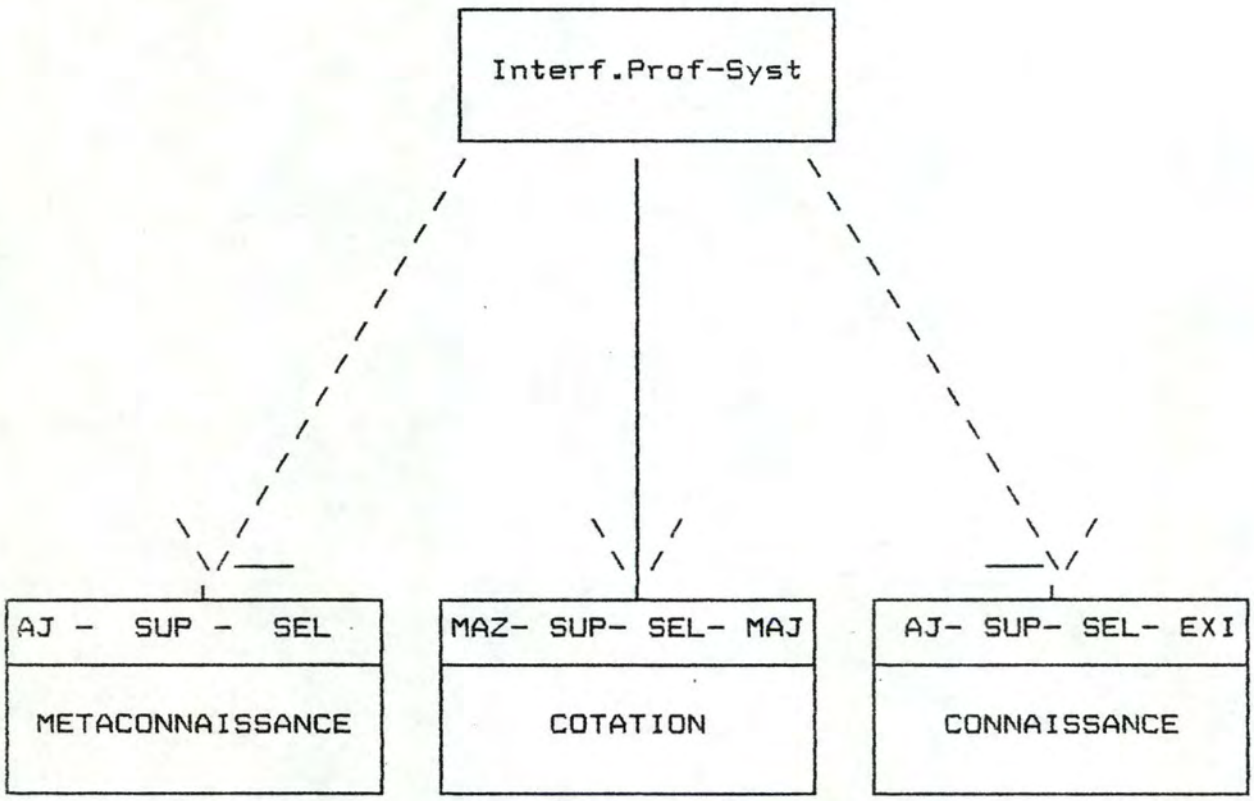
ET il n'existe pas de sous-système utile incluant A et non B.

Un **sous-système utile**, c'est un ensemble de modules autonomes.

## IV.1 LES DEUX APPLICATIONS

**Application 1 : Introduction de la matière.**

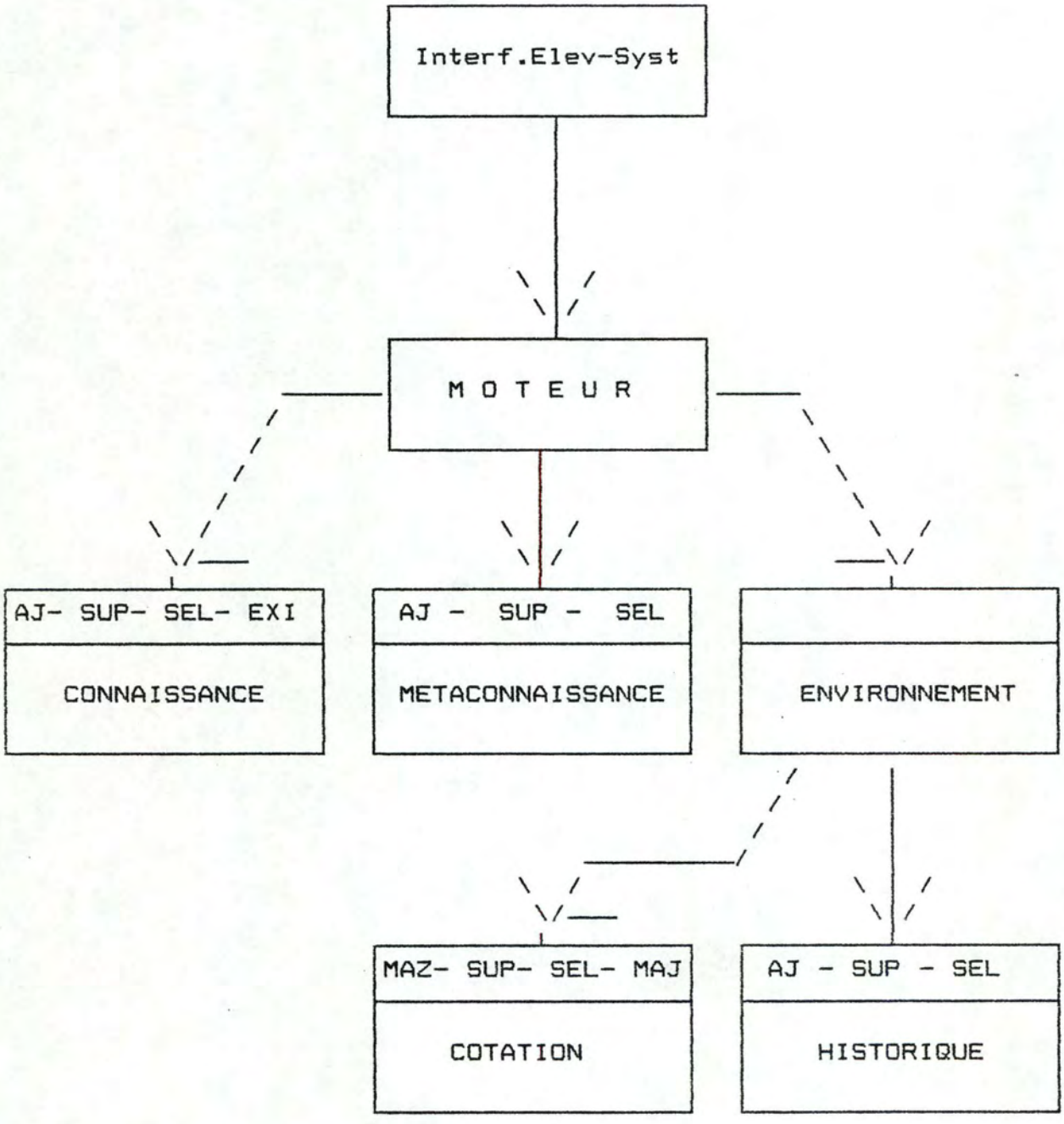
**OBJECTIF :** Le professeur va introduire la matière que l'élève devra mémoriser, c'est-à-dire introduire la connaissance et la métaconnaissance.




Architecture modulaire de relation utilise pour l'application 1

**Application 2 : Apprentissage.**

**OBJECTIF:** L'élève va maintenant s'efforcer de mémoriser la matière en étant aidé par le système.




Architecture modulaire de relation utilise pour la application 2

Nous allons maintenant définir les différents modules utilisés. Pour spécifier les procédures, nous emploierons la méthode des assertions : préconditions et postconditions.

Pour chaque procédure, nous donnerons son nom, la liste des paramètres (arguments et résultats), ainsi que son objectif.

Nous donnerons également le ou les préconditions (caractérisant les propriétés à satisfaire avant l'exécution de la procédure pour que celle-ci s'exécute correctement) et le ou les postconditions (conditions explicitant les propriétés des résultats de la procédure, qui doivent être satisfaites après chaque exécution correcte).

## IV.2 Le module Connaissance

Ce module est un module de données. Celles-si sont des occurrences de relation; elles seront classées par ordre alphabétique, par relation, et pour une même relation, séquentiellement.

Il sera accessible à partir des primitives suivantes :

**1 AJCO ( nomderel, listeitems )**

Objectif : La procédure a pour objectif de rajouter à la connaissance l'occurrence de relation **nomderel** et la liste de ses items associés **listeitems.**

Précond : Nomderel et listeitems sont non vides.

Postcond : L'occurrence est ajoutée à la connaissance.

**2 SELCO ( nomderel, listeitems )**

Objectif : La procédure a pour objectif de sélectionner l' occurrence de relation suivante de nom **nomderel** qu'elle rencontre dans la connaissance et de placer la liste des valeurs d'items dans **listeitems.**

Précond : Nomderel n'est pas vide et existe dans la connaissance.

Postcond : Listeitems est instanciée.

**3 EXISTECO (Nomderel, réponse)**

Objectif : La procédure regarde si une occurrence de la relation de nom **nomderel** existe dans la connaissance. **Réponse** est vrai ou faux.

Précond : Nomderel est non vide.
Postcond : Réponse est vrai si il existe déjà et faux dans le cas contraire.

4 **SUPCO ( nomderel, listeitems )**

Objectif : La procédure a pour objectif de supprimer de la connaissance l'occurrence de relation **nomderel** et la liste de ses items associés **listeitems.**

Précond : Nomderel et listeitems sont non vides.

Postcond : La relation nomderel, ses items et le niveau de connaissance associé sont enlevés de la connaissance **ou**
aucune incidence si la relation ou l'occurrence n'existe pas.

5 **SELPREMCO ( nomderel, listeitems )**

Objectif : La procédure a pour objectif de sélectionner la première occurrence de relation de nom **nomderel** et de placer la liste des valeurs d'items dans **listeitems.**

Précond : Nomderel n'est pas vide et existe dans la connaissance.

Postcond : Listeitems est instanciée.

## IV.3 Le module Métaconnaissance

Ce module est un module de données qui sont des règles, classées dans l'ordre d'ajout.

Une règle est soit
- une information à donner à l'élève et sa condition d'application.
- une information à accepter de l'élève et sa condition d'acceptation.
- une question à poser à l'élève et sa condition d'application.
- une question à accepter de l'élève et sa condition d'acceptation.

Le module sera accessible à partir des primitives suivantes :

**1 AJCMETACO (règle)**

Objectif : La procédure a pour objectif de rajouter au fichier métaconnaissance la règle **règle.**

Précond : Règle est non vide.

Postcond : Règle est ajoutée à la métaconnaissance.

**2 SELMETACO (règle)**

Objectif : La procédure à pour objectif de sélectionner la première règle qu'elle rencontre dans le fichier métaconnaissance et de la placer dans **règle.**

Précond : La métaconnaissance n'est pas vide.

Postcond : Règle est instanciée.

3 SUPMETACO (règle)

Objectif : La procédure a pour objectif de supprimer de la métaconnaissance la règle **règle.**

Précond : Règle est non vide.

Postcond : Règle est enlevée de la métaconnaissance.

4 SELPREMMETACO (règle)

Objectif : La procédure a pour objectif de sélectionner la première règle de la métaconnaissance dans le fichier métaconnaissance et de la placer dans **règle.**

Précond : La métaconnaissance n'est pas vide.

Postcond : Règle est instanciée.

## IV.4 Le module Historique

Ce module est un module de données qui sera accessible à partir des primitives suivantes :

**1 AJHIS ( question,réponse,date,heure,ok )**

Objectif : La procédure a pour objectif de rajouter à l'historique la question **question** et la réponse **réponse** fournie par l'élève à telle **date**, telle **heure.**

Précond : Question, réponse,date,heure sont non vides.

Postcond : Le tout est ajouté à l'historique et ok est vrai. Ok est faux s' il exsite déja 5 questions dans l'historique; l'ajout n'est alors pas fait.

**2 SELHIS ( question, réponse, date, heure )**

Objectif : La procédure a pour objectif de sélectionner la première occurrence qu'elle rencontre dans l'historique et de complèter les valeurs de **question,réponse,date,heure.**

Précond :

Postcond : Les paramètres ont une valeur si l'historique n'est pas vide.

**3 SUPHIS ( question,réponse,date,heure )**

Objectif : La procédure a pour objectif de supprimer de l'historique l'occurrence déterminée par **question, réponse, date, heure.**

Précond : Question,réponse,date et heure sont non-vides.

Postcond : L'occurrence est enlevée de l'historique si elle existe

## IV.5 Le module Cotation

Ce module est un module de données qui associe à chaque occurrence de relation de la connaissance une cote correspondant à l'évaluation de la connaissance de l'élève pour cette occurrence.

Il sera accessible à partir de primitives suivantes :

**1 MAZ ( idocc )**

Objectif : La procédure a pour objectif de donner la valeur 0 à l'occurrence de relation identifiée par **idocc.**

Précond : Idocc correspond à une et une seule occurrence de relation de la connaissance.

Postcond : A cette occurrence est associée une cote qui a la valeur 0.

**2 MAJ ( idocc, cote)**

Objectif : La procédure a pour objectif de donner la valeur **cote** à l'occurrence de relation identifiée par **idocc.**

Précond : Idocc correspond à une et une seule occurrence de relation de la connaissance.
cote >= 0

Postcond : A cette occurrence est associée la cote **cote.**

**3 SELCOTE ( idocc, cote )**

Objectif : La procédure a pour objectif de donner à **cote** la valeur associée a l'occurrence de relation de la connaissance identifiée par **idocc.**

Précond : Idocc correspond à une et une seule occurrence de relation de la connaissance.

Postcond : Cote prend sa valeur.

**4 SUPCOTE ( idocc)**

Objectif : La procédure supprime la cote associée à l'occurrence de relation de la connaissance identifiée par **idocc**.

Précond : Idocc identifie une occurrence.
Postcond : La cote associée est supprimée.

## IV.6 Le module Moteur

Ce module est le module de traitement central de l'application 2. Il sera appelé par l'interface élève-système et utilisera les modules de connaissance, de métaconnaissance et d'environnement.

Il sera accessible de la façon suivante:

**MOTEUR( Qàposer, Qàrecev, Iàdonn, Iàrecev )**

Objectif : Le moteur, à partir d'une information fournie par l'élève **Iàrecev** ou d'une question qu'il a posé **Qàrecev**, va décider de la prochaine action qu'il fera : soit poser une question **Qàposer**, soit donner une information **Iàdonn**.

Arg : Qàrecev, Iàrecev

Précond : Soit Qàrecev soit Iàrecev a une valeur.

Résultats: Qàposer, Iàdonn

Postcond : Soit Qàposer, soit Iadonn a une valeur.

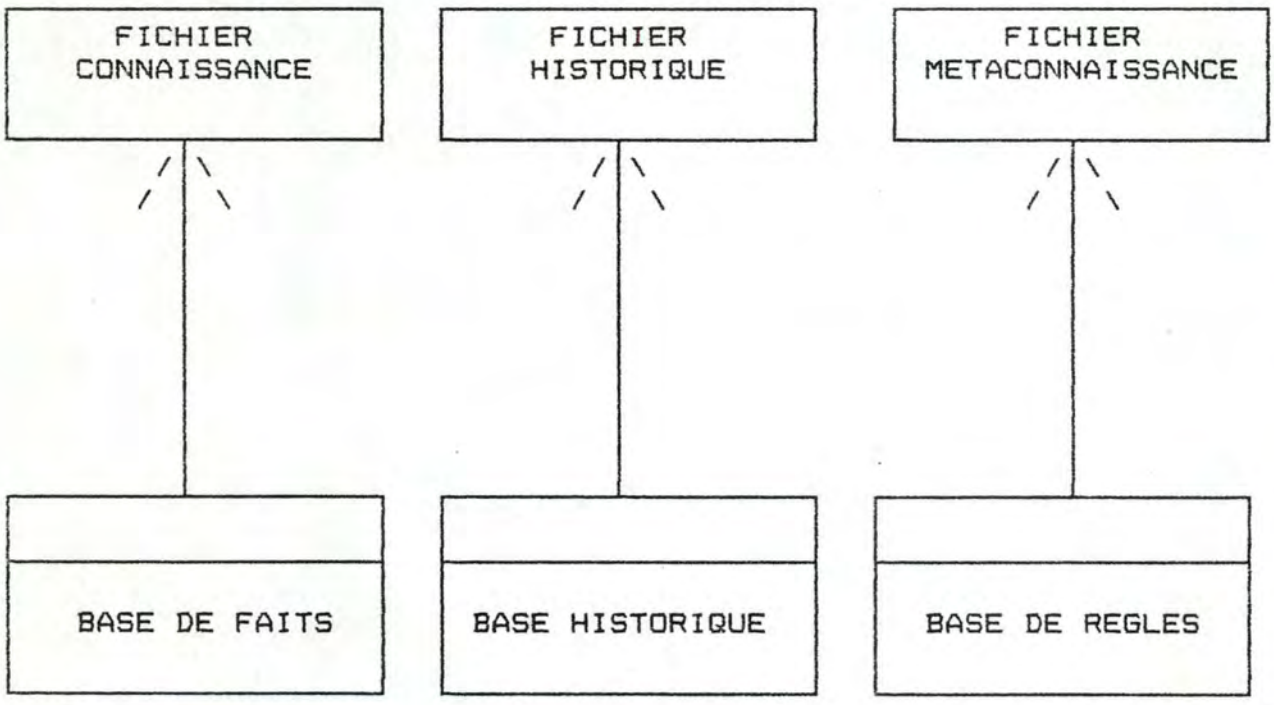
## IV.7 Représentations interne et externe

Le système utilisera une représentation de la connaissance, de la métaconnaissance et de l'historique directement à partir de la mémoire centrale.

Il faudra donc prévoir des modules qui transforment ces représentations de la mémoire de l'ordinateur vers une mémoire auxiliaire et réciproquement.

### IV.7.1 Sauvetage de la mémoire centrale.

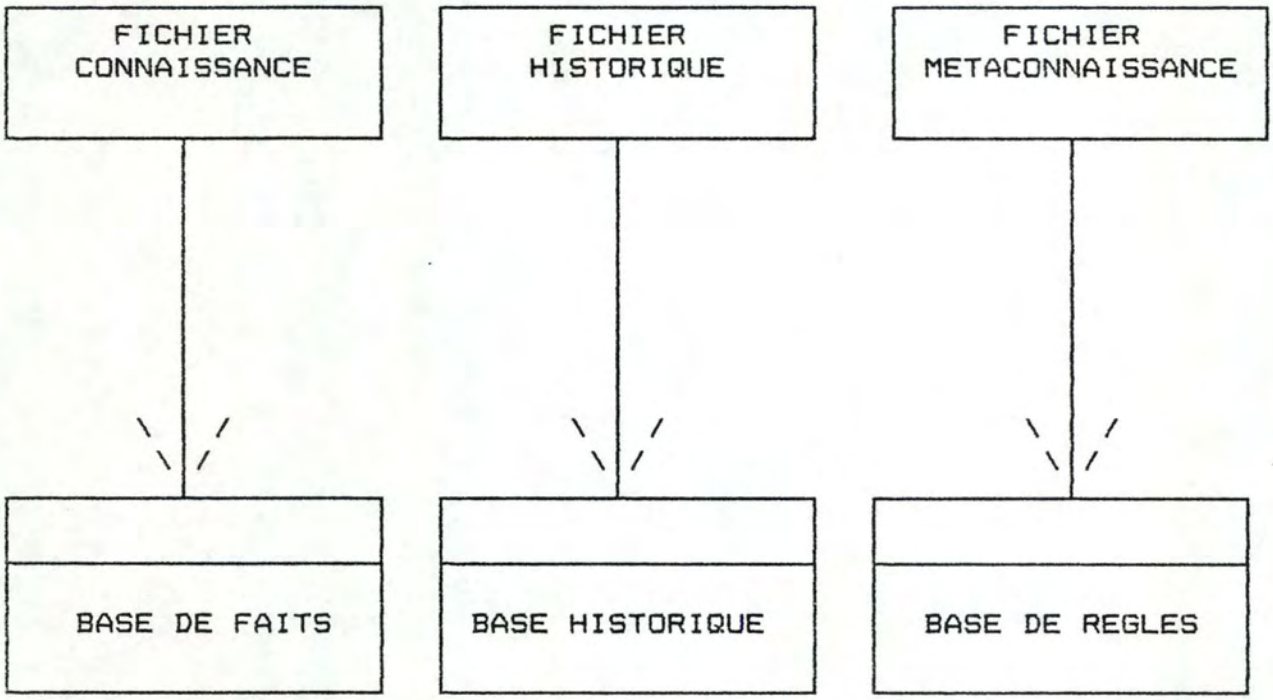
**OBJECTIF:** Il faut sauvegarder connaissance, métaconnaissance et historique sur support secondaire.




Architecture modulaire de relation utilise pour la phase de sauvetage

## IV.7.2 Restauration en mémoire centrale

**OBJECTIF:** Il faut restaurer connaissance, métaconnaissance et historique en mémoire centrale.




Architecture modulaire de relation utilise pour la phase de restauratio

### IV.7.3 Nom des primitives d'appel des modules

#### IV.7.3.1 La phase de sauvetage

**nommat = nom de matière**
**nomélève = nom d'élève**
**date = date du jour**
**heure = heure actuelle**

**SAVEFICHCONN ( nommat, noméleve, date, heure )**

pour le module fichier connaissance

**SAVEFICHHIST ( nommat, noméleve, date, heure )**

pour le fichier historique

**SAVEFICHMETA ( nommat, date, heure )**

pour le fichier métaconnaissance

#### IV.7.3.2 La phase de restauration

**nommat = nom de matière**
**nomélève = nom d'élève**
**date = date du jour du dernier sauvetage**
**heure = heure du dernier sauvetage**

**RESTFICHCONN ( nommat, noméleve, date, heure )**

pour le module fichier connaissance

**RESTFICHHIST ( nommat, noméleve, date, heure )**

pour le fichier historique

**RESTFICHMETA ( nommat, date, heure )**

pour le fichier métaconnaissance

# CHAPITRE V : IMPLEMENTATION D'UNE MAQUETTE.

## V.1 Choix de l'exemple et justification

Il faut, avant tout, rappeler que le système se distingue des didacticiels traditionnels par description synthétique de la matière qu'il permet.

C'est donc un gain de temps important offert au professeur qui ne doit donner qu'une seule fois la même question dans un même programme.

Dans cette optique, nous avons choisi de développer une maquette d'aide à la mémorisation des tables de multiplications et d'additions. En effet, il sera également possible de définir la connaissance très facilement puisque les ordinateurs connaissent déjà la représentation de ces notions.

Cela simplifiera très fort la définition de la connaissance à faire mémoriser puisque nous pourrons profiter de cette représentation.

Faisons toutefois attention à la remarque suivante.

Nous avions dit que le système ne permettrait pas d'attirer l'attention de l'élève sur des "trucs" de mémorisation ou même des algorithmes de déduction. Or, certains pourraient objecter que l'arithmétique est typiquement du calcul et donc pas de la mémorisation. Cela n'est pas vraiment exact. Au début, un enfant ne sait pas que 1+1=2, il faut qu'il le mémorise.

Maintenant, il est vrai que, connaissant l'addition correctement, on peut retrouver toutes les multiplications.

Notre mémoire a toutefois mémorisé ce genre de calcul. Quand nous demandons combien font 8 x 8 à un adulte, il répondra immédiatement 64 et ne devra pas calculer huit fois l'addition de 8.

Ceci nous permet donc de conclure que, effectivement, même s'il existe des "trucs" permettant de retrouver des notions mémorisées, ils n'apportent pas nécessairement le chemin le plus rapide et la mémorisation s'avère parfois obligatoire.

L'exemple choisi nous permet de montrer l'utilité de chacun des concepts relevés dans les chapitres précédents. Nous pourrons, ainsi, faire fonctionner les différents modules.

## V.2 Spécification de la matière

Nous allons dans ce paragraphe spécifier la matière que nous voulons définir pour la maquette.

### V.2.1 Spécification de la connaissance

Comme nous l'avons fait remarquer, la connaissance à faire mémoriser peut être facilement et naturellement représentée sous forme de schéma relationnel. Nous procéderons donc de cette façon.

```
plus(X,Y,X+Y).
fois(X,Y,XxY).
```

Nous avons ainsi décrit synthétiquement les deux tables.

Etant donné que l'ordinateur possède déjà une représentation interne pour les nombres entiers, nous pouvons définir le range des opérations à faire mémoriser.

par exemple :

**X<10.**

**Y<10.**

### V.2.2 Spécification de la métaconnaissance

Il faut définir plusieurs règles pour le système. Rappelons que la syntaxe utilisée est celle du langage Prolog.

#### V.2.2.1 Le seuil de connaissance.

A chaque relation sera associée une cote. Celle-ci représente le pourcentage de points que l'élève doit atteindre pour qu'il soit considéré par le système comme ayant mémorisé la relation.

**plusconnu:-Cote>8.**

Nous définissons ainsi que la relation plus est connue si l'élève a obtenu au moins 9 à chacune des occurrences de cette relation plus. Nous faisons de même pour la relation fois.

**foisconnu:-Cote>8.**

### V.2.2.2 Les informations à donner (iad)

Il faut donner au système la manière dont il doit présenter la connaissance.

```
iad:-plus(X,Y,Z),write(X," plus ",Y," égale ",Z).
iad:-fois(X,Y,Z),write(X," fois ",Y," égale ",Z).
```

Le mot iad indique qu'il s'agit d'une information que le système peut donner.

Celle-ci porte sur la relation "plus" dans le premier cas, "fois" dans le second. Est ensuite décrite la façon dont toutes les occurences de la relation définie seront présentées.

### V.2.2.3 Les questions à poser (qap)

```
qap:-plus(X,Y,Z),Cote<9,write(X," plus ",Y," égale ?").
qap:-fois(X,Y,Z),Cote<9,write(X," fois ",Y," égale ?").
```

"Qap" indique qu'il s'agit d'une question que le système peut poser à l'élève. Prenons la première régle : elle porte sur la relation plus, il ne faut pas poser de questions sur une occurence qui aurait une cote supérieure à 9; si c'est la cas, la façon de poser la question est donnée dans le write.

### V.2.2.4 Les informations à accepter (iaa)

```
iaa:-plus(X,Y,Z),Y<5,Z<5.
iaa:-fois(X,Y,Z),Y<5,Z<5.
```

En définissant ces deux règles, nous disons au système que l'élève pourra donner des informations portant sur les relations plus et fois où les valeurs des deux premiers paramètres sont inférieures à 5.

### V.2.2.5 Les questions à accepter (qaa)

```
qaa:-plus(X,Y,?),Y<5,Z<5.
qaa:-fois(X,Y,?),Y<5,Z<5.
```

C'est ainsi que le professeur dira au système que l'élève peut poser des questions sur plus et moins si les X e Y sont inférieurs à 5.

### V.2.2.6 Les messages d'erreurs

C'est par ce biais que le professeur pourrait signaler au système des erreurs typiques de ce genre de matière et les informations à donner alors à l'élève.

```
eval(Z,Rep):-X<>Y,write("c'est ",Z," la bonne réponse").
eval(Z,Rep):-X=Y,write("c'est bon").
```

Le professeur donne comme paramètre à "eval", Z, la bonne réponse, et Rep, la réponse de l'élève. Il peut ajuster les messages à faire apparaître comme il le veut. Nous en avons choisi deux simples si X = Rep et si X <> Rep.

V.2.2.7 Les notions pédagogiques.

Nous souhaitons que la mémorisation se fasse en deux phases. Nous trouvons qu'il est préférable pour l'élève de connaître à fond la relation "plus" avant de mémoriser la relation "fois". Nous pouvons définir ceci en ajustant l'iad et la qap portant sur fois.

Il suffit de rajouter la condition plusconnu.

```
qap:-fois(X,Y,Z),plusconnu,Cote<9,write(X," fois ",Y," égale ?").
iad:-fois(X,Y,Z),plusconnu,write(X," fois ",Y," égale ",Z).
```

## V.3 Spécification de la maquette

### V.3.1 Objectif

L'objectif de ce programme est de permettre d'aider un élève à la mémorisation des tables d'additions et de multiplications. Celles-ci seront définies synthétiquement par le professeur. Il décrira aussi un ensemble de règles permettant au système de savoir comment présenter la connaissance à faire mémoriser à l'élève, comment lui poser des questions, quelles sont les questions et les informations que l'élève peut fournir. Enfin le professeur indiquera les messages à donner à l'élève suivant qu'il répond correctement ou non.

Il n'y aura qu'une seule bonne réponse possible par question.

La connaissance à faire mémoriser et les règles seront définies comme indiqué dans le paragraphe 2 ci-dessus.

Lors de la mémorisation, outre les règles définies par le professeur, le système respectera aussi les points suivants :

- Ne pas présenter plus de 7 nouvelles informations en une fois.
- Ménager une pause toutes les quarantes minutes.
- Utiliser le système d'évaluation présenté dans ce travail.

L'élève pourra, à sa guise, poser des questions permises par le professeur, donner des informations permises et apprendre des informations considérées comme non mémorisées.

Il pourra aussi s'arrêter quand il le souhaite, le système mémorisera alors le niveau de connaissance.

### V.3.2 Interface professeur-système

Un menu proposera au professeur de sélectionner un des 3 points suivants: - insertion de connaissance

- insertion de connaissance de façon synthétique
- insertion de règle de métaconnaissance

#### V.3.2.1 Insertion de connaissance

Le professeur donnera le nom de relation et la liste des items de l'occurence.

exemple: plus(2,2,4)

#### V.3.2.2 Description synthétique

Elle sera donnée comme présentée ci-dessus.

#### V.3.2.3 Description des règles.

Une règle commence par **iad**, **qap**, **qaa**, **iaa**, **eval**, ou <mon de relation>**connu**.
Elle est suivie par :- et la suite de la règle.

Pour une définition exacte de la description de la métaconnaissance de cette maquette, reportez-vous ci-dessus.

Remarque: Il est très important de faire attention à l'emploi des minuscules et majuscules, celles-ci ayant leur signification. Toutes les lettres doivent être minuscules, sauf la première d'un paramètre, qui doit être majuscule.

### V.3.3 Interface élève-système.

Sur son écran , l'élève verra aparaître plusieurs fenêtres: une pour les informations à donner, une pour les questions à poser, une pour les informations à accepter et une pour les questions à accepter. C'est l'endroit où se placera le curseur qui déterminera ce que peut faire l'élève.

#### V.3.3.1 Comment poser des questions au système?

L'élève donnera un nom de relation et les propositions qu'il fait avec un ? en 3ème item.

Exemple: plus(2,2,?)

#### V.3.3.2 Comment informer le système?

Il donnera un nom de relation et les propositions qu'il fait.

Exemple: plus(2,2,4)

#### V.3.3.3 Comment répondre au système?

Il suffira de donner le nombre mémorisé.

Exemple: **2 plus 2 =** 4

En gras, la question du système.

### V.3.4 Remarque sur l'interface

Nous sommes conscient que cet interface n'est pas très convivial, mais notre but n'est pas, ici, de faire un beau programme, mais bien de montrer qu'il est possible de développer un système d'aide à la mémorisation.

## V.4 Choix du langage et du matériel

### V.4.1 Choix du matériel

Comme il était techniquement possible de développer cette maquette sur un micro-ordinateur, nous avons décidé d'adopter ce genre de machine.

En effet, ils offrent de nombreux avantages. Parmi ceux-ci, nous pouvons relever, le fait que, actuellement, les différents établissements scolaires sont équipés avec ce genre de matériel.

Cependant, en ce domaine, les équipements sont différents selon les écoles scolaires et les marques d'ordinateur utilisées sont diverses: Apple, Commodore, Spectrum... Elles sont, hélas, bien souvent incompatibles.

Toutefois, ces dernières années ont vu proliférer les PC compatibles et leur prix est devenu relativement abordable. C'est pour cette raison qu'ils ont fait leur apparition dans les écoles et gageons que, grâce au standard apporté, ils envahiront de plus en plus le marché de l'enseignement.

C'est pourquoi nous avons choisi cet appareil pour le développement de notre maquette.

### V.4.2 Choix du langage de programmation

Le langage d'implémentation de cette maquette est le langage turbo-prolog de Borland (Cfr référence 3).

Nous l'avons choisi car, d'une part, il permet de définir la connaissance à mémoriser directement sous forme relationnelle, et car, d'autre part, il permet une représentation des règles de la métaconnaissance où les conditions d'application sont directement jointes aux actions à accomplir.

## V.5 Architecture logicielle de la maquette

La maquette comprend deux applications: l'une pour le professeur, l'autre pour l'élève.

L'architecture modulaire de cette maquette est la même que celle proposée pour le système général. Elle comprend les modules métaconnaissance et connaissance, qui sont utilisés par l'application professeur. A la fin de cette application, deux textes ont été crées, la connaissance et la métaconnaissance. Ils ont tout deux une forme interprétable par le langage prolog.

L'application élève utilise les modules cotation et historique ainsi que les textes de connaissance et métaconnaissance résultant de l'application professeur.

Voici, pour chacun des modules, les primitives dévelopées.

### V.5.1 Module connaissance

**AJCO** : Ajouter de la connaissance

**AJCOSYNT** : Ajouter de la connaissance de façon synthétique.

### V.5.2 Module métaconnaissance

**AJMETACO** : Ajouter des règles de métaconnaissance

### V.5.3 Module Cotation

**MAJCOTE** : Mise à jour de la cote d'une occurence de relation

### V.5.4 Module historique

**AJHIST** : Ajout d'une question et de la réponse reçue à l'historique. Maximum 5 couples questions-réponses sont gardés, les 5 plus récents.

### V.5.5 L'application professeur

Celle-ci présente le menu au professeur, sélectionne son choix et lance la primitive correspondante.

### V.5.6 L'application élève

Le moteur de la mémorisation gère le tout. Il permet d'accepter ce que veut l'élève : poser des questions, donner des informations, mémoriser de la nouvelle connaissance.

Il utilise les règles définies dans la métaconnaissance.

## V.6 Manuels utilisateurs de la maquette

Nous allons maintenant proposer les manuels pour l'utilisation de la maquette. Nous commençons par celui du professeur et nous donnerons ensuite celui de l'élève.

### V.6.1 Manuel du professeur

Lorsque vous voulez lancer le programme, allumez l'ordinateur en y placant votre disquette dans le drive principal. Tapez, ensuite, la date et l'heure comme dans l'exemple ci dessous.

Ensuite, taper "**prof**" et <**return**>. Vous avez à ce moment lancé le programme destiné au professeur. L'écran suivant apparaît :

PROFESSEUR

```
1...Ajout de fragments de connaissance
2...Ajout de règles de métaconnaissance
3...Ajout de connaissance de façon synthétique.une à la fois
4...Fin de travail



Votre choix:
```

Vous devez maintenant entrer la matière à faire mémoriser. Entrez le numéro de votre choix et tapez return.

### V.6.1.1 Vous avez entré le chiffre 1

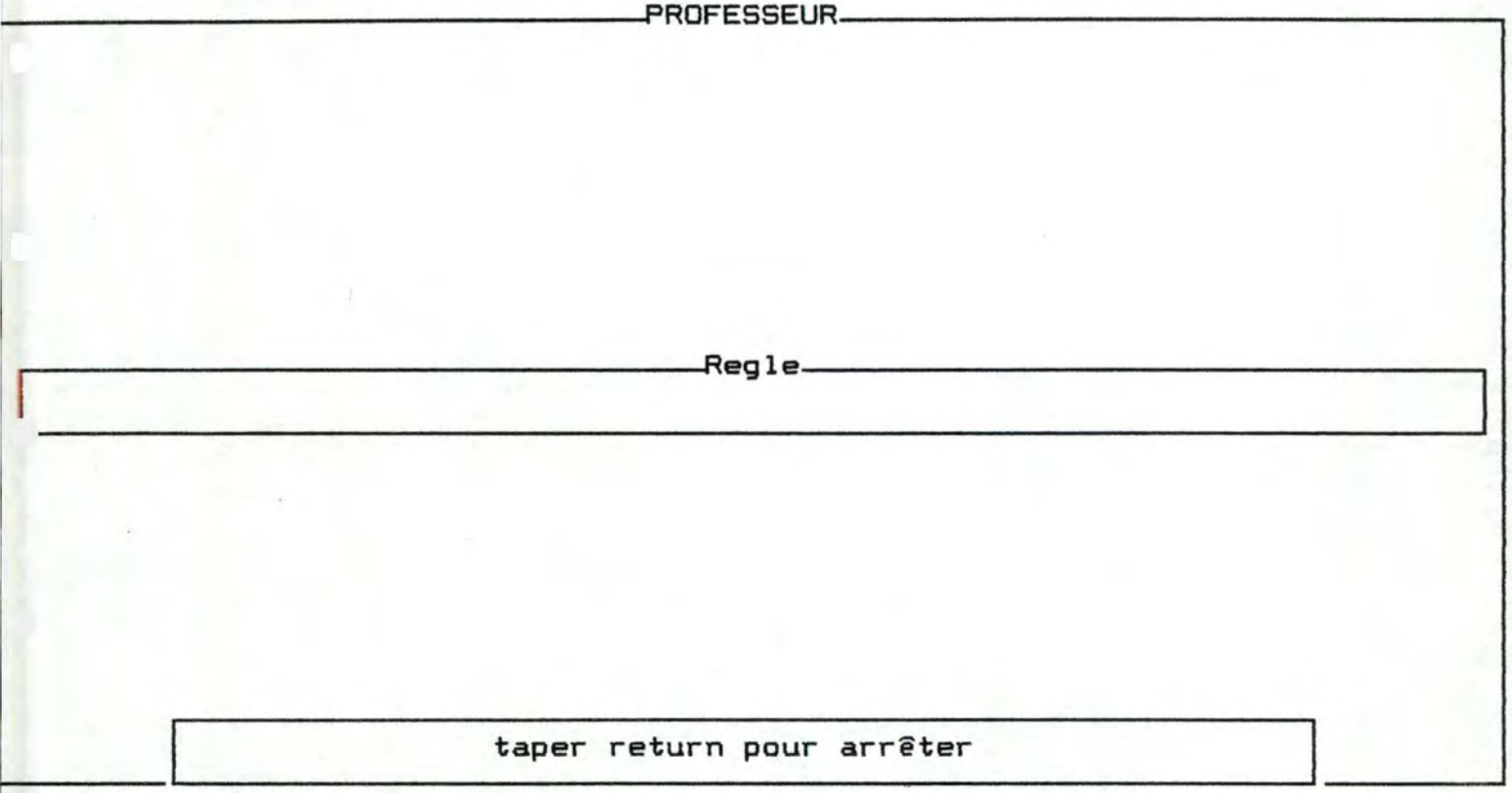
L'écran ci-dessous aparaît. Vous entrez alors une occurence de relation et <Return>. Vous faites cela pour toutes les occurences et, quand vous avez fini, tapez encore <Return>.

PROFESSEUR

La nouvelle occurence de relation:

### V.6.1.2 Vous avez entré le chiffre 2

L'écran ci-dessous apparaît. Vous entrez alors une règle complète et <Return>. Vous faites cela pour toutes les règles de la métaconnaissance. Quand vous avez fini, tapez encore <Return>.




### V.6.1.3 Vous avez entré le chiffre 3

L'écran ci-dessous apparaît. Vous entrez alors une à une les instructions décrivant la connaissance et <Return>. Quand vous avez fini, tapez encore <Return>.

PROFESSEUR

La nouvelle description de relation:

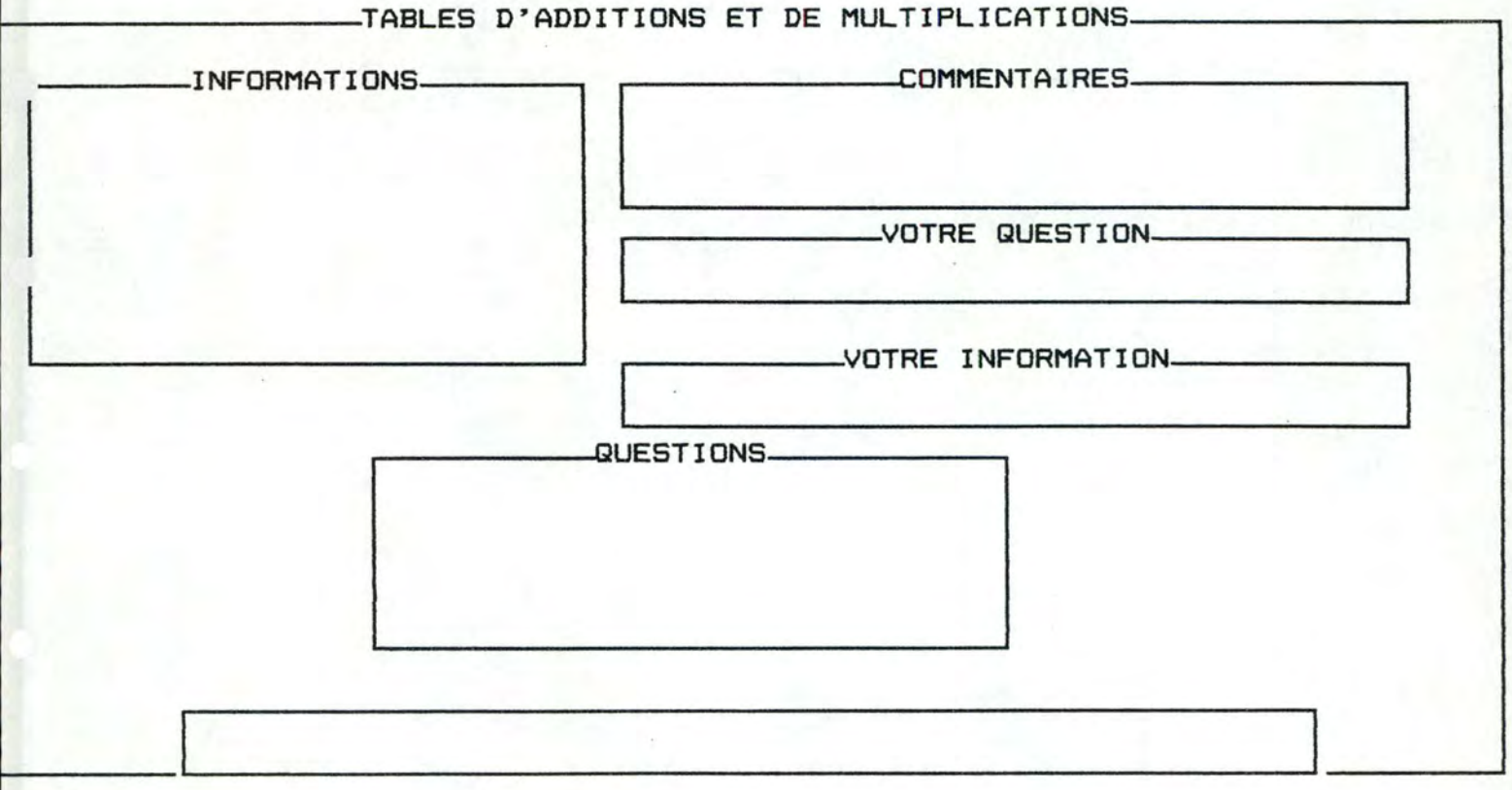
### V.6.1.4 Vous avez entré le chiffre 4

C'est l'option à choisir quand vous avez définit toute la matière. Votre système est maintenant prêt pour aider l'élève à mémoriser.

### V.6.2 Manuel de l'élève

Lorsque vous voulez lancer le programme, allumez l'ordinateur en y plaçant votre disquette dans le drive principal. Tapez, ensuite, la date et l'heure comme dans l'exemple ci dessous.

Ensuite, taper "**élève**" et <**return**>. Vous avez, à ce moment, lancé le programme destiné a l'élève. L'écran suivant apparaît :



Vous voyez devant vous le système d'aide à la mémorisation des tables d'additions et de multiplications. Vous pouvez à tout moment choisir entre :

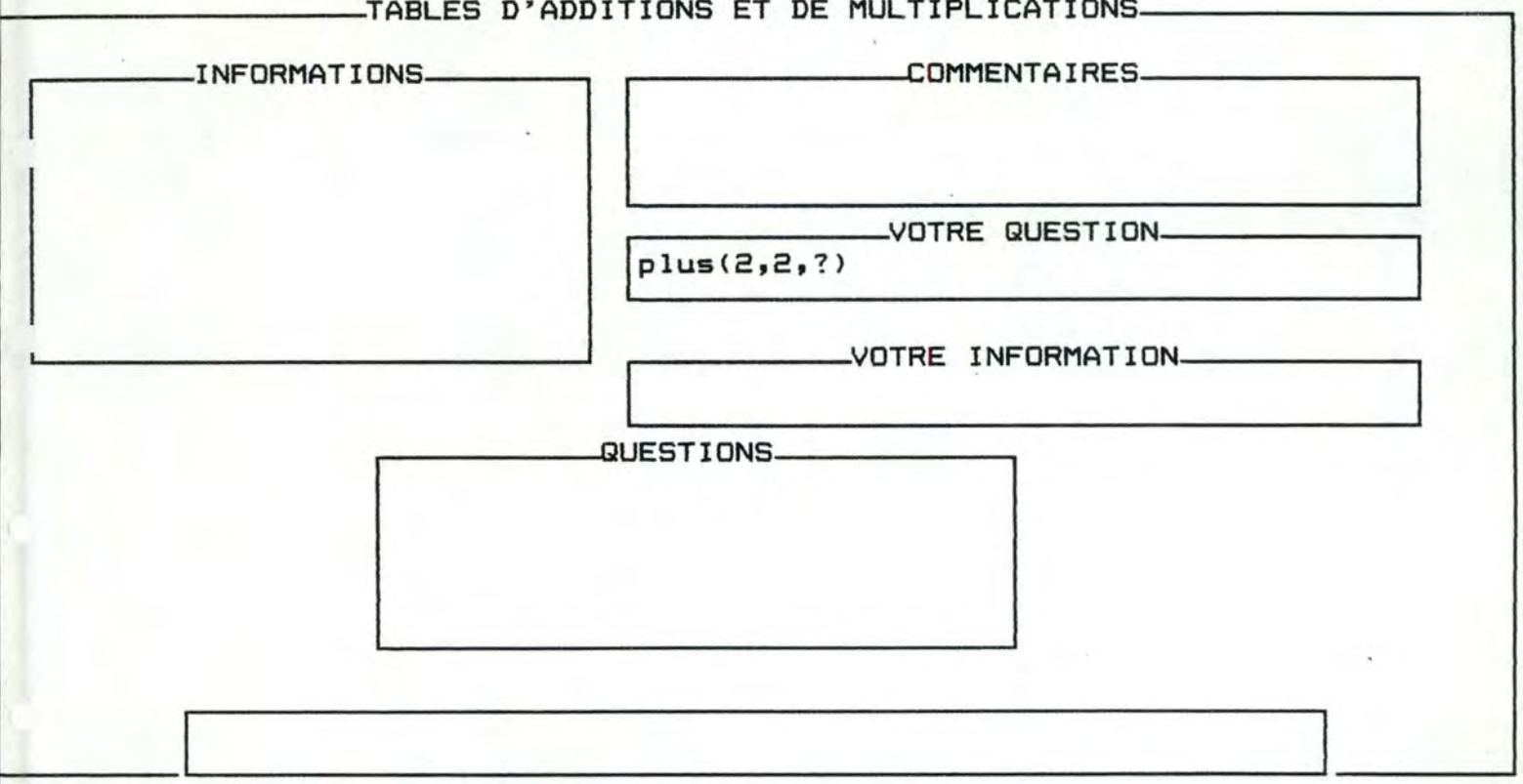
1.. poser une question sur la matière

2.. donner une information sur ce que vous connaissez déjà.

3.. voir de la matière non connue

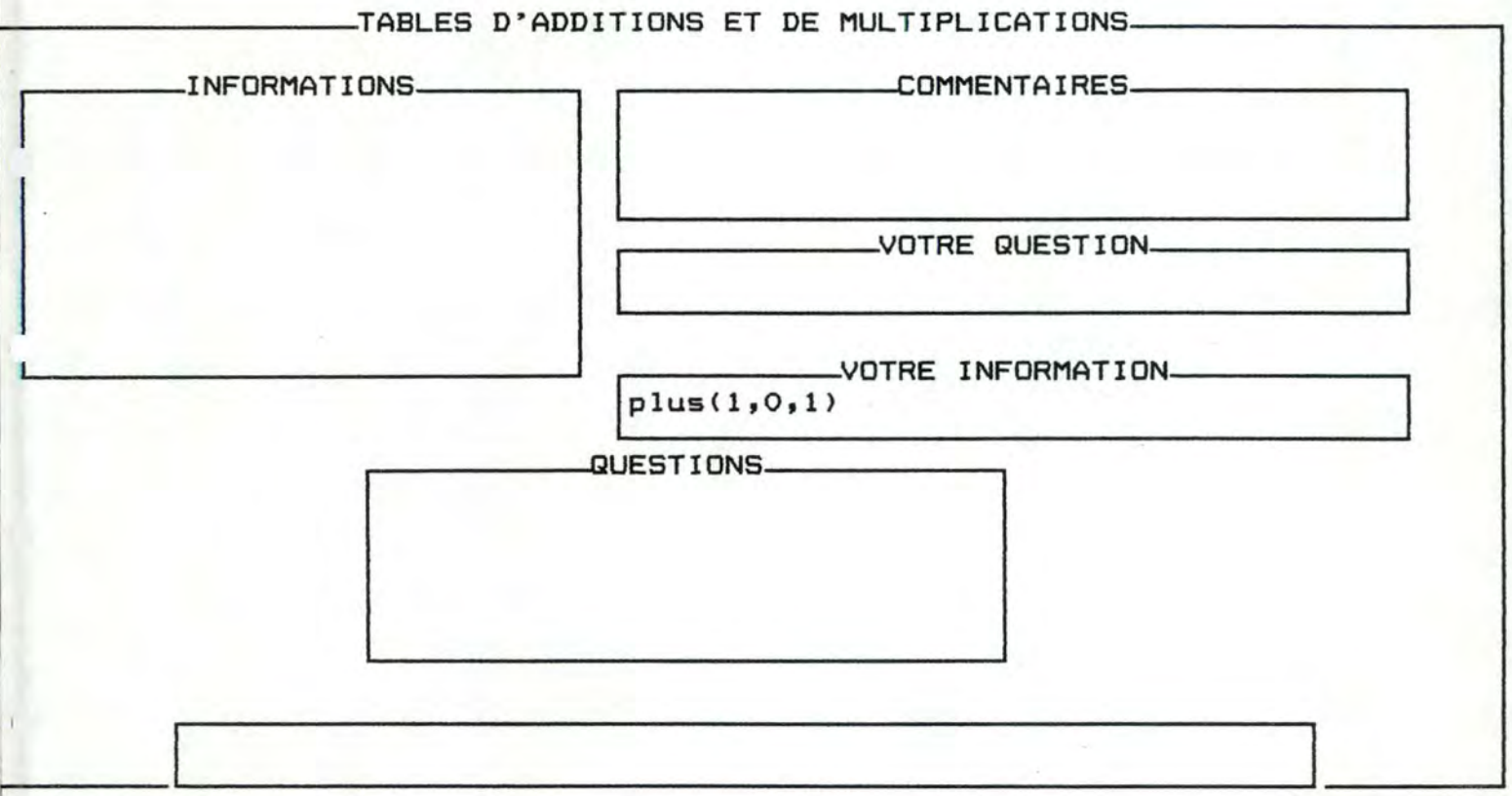
4.. répondre à des questions sur la matière que vous devez mémoriser.

C'est l'endroit où se trouve le curseur qui détermine ce que vous voulez faire. Au début, il se trouve dans la fenêtre intitulée "Votre Question". Vous pouvez, alors, soit poser une question comme dans l'écran ci-dessous, soit taper <Return> pour aller dans la fenêtre intitulée "Votre Information".



**Vous êtes dans la fenêtre "Votre Information"**

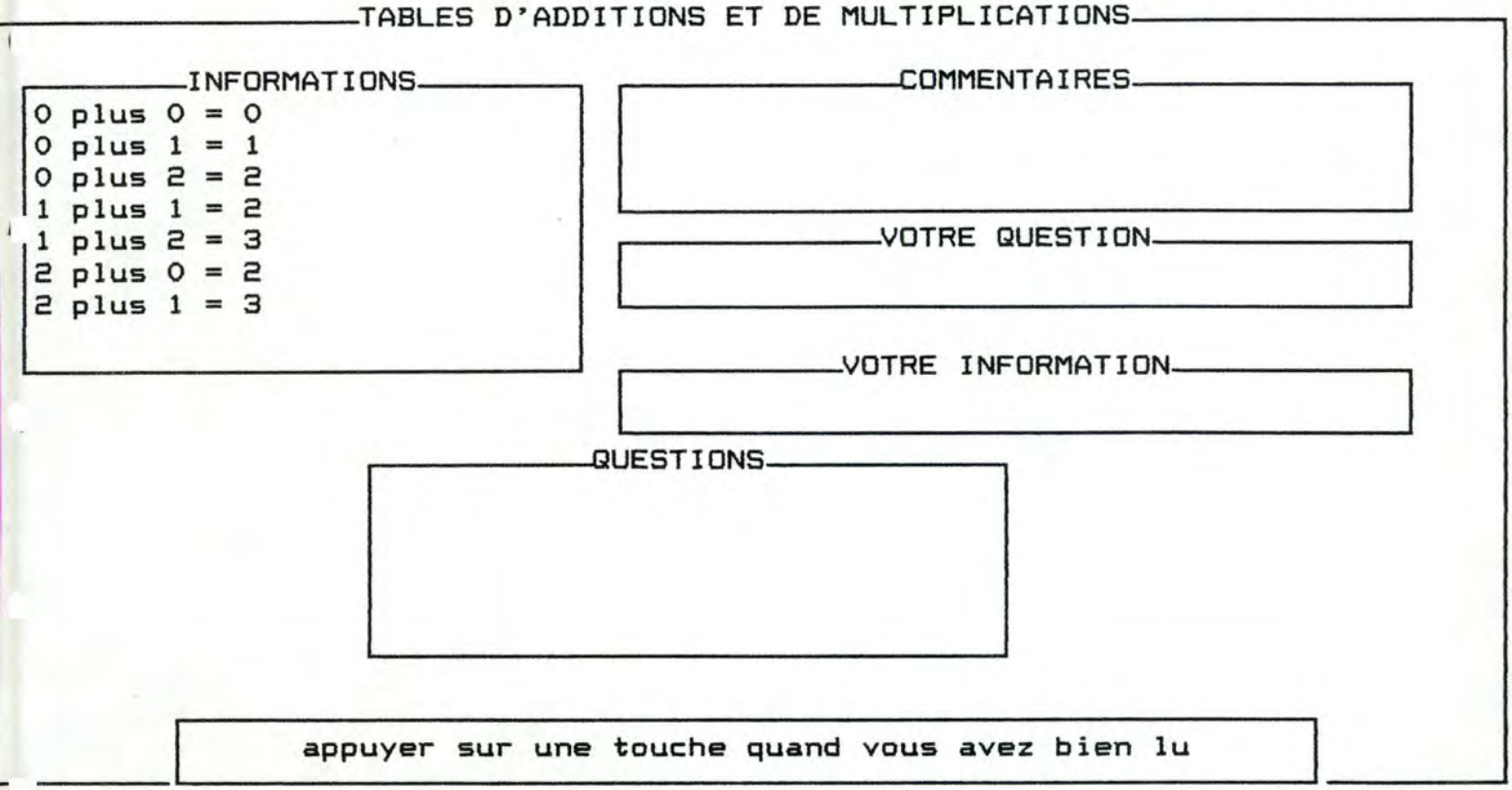
Vous pouvez, alors, soit donner une information comme dans l'écran ci-dessous, soit taper <Return> pour voir de la matière non mémorisée.




**Vous êtes dans la fenêtre "Informations"**

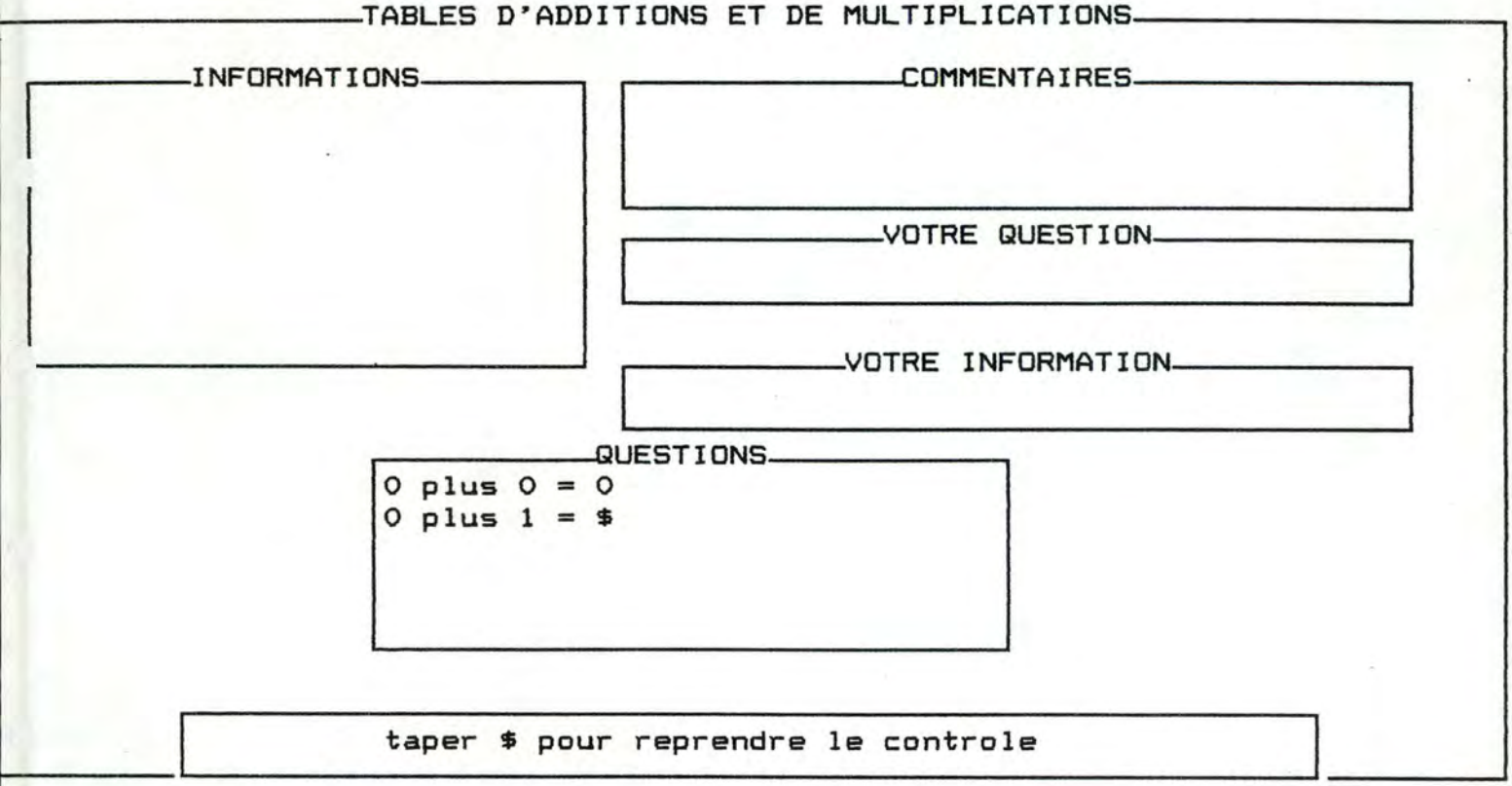
La nouvelle matière vient de s'afficher, lisez-la attentivement et essayez de la retenir. Quand c'est fait, tapez <Return>.

Le système va alors vous poser des questions pour voir si vous connaissez suffisamment. Vous êtes passé dans la fenêtre "Questions".




**Vous êtes dans la fenêtre "Questions"**

Répondez alors aux questions. Si vous souhaitez revenir à un des points précédents, tapez "**$**" à une question. Vous pourrez alors, soit quitter le système, soit revenir dans la fenêtre intitulée "Votre question".



# CHAPITRE VI : Les limites du travail

Comme tout travail, celui-ci possède des limites. Nous avons choisi de les développer dans un chapitre particulier pour bien montrer l'importance que nous y attachons, estimant qu'elles sont essentielles à la maîtrise du système.

En effet, tout système, qu'il soit informatique ou autre, ne peut être considéré comme maîtrisé que si ses limites sont connues.

Nous relèverons plusieurs limites à ce travail: la base théorique sous-jacente (VI.1), l'utilisation d'une maquette (VI.2), les outils d'interface du système (VI.3), le type d'association "se sert de" (VI.4) et l'expérimentation (VI.5).

## VI.1 La base théorique

### VI.1.1 La structure de représentation de la mémoire humaine

Comme nous l'avons déjà souligné, les sciences humaines ne nous fournissent pas actuellement de théorie unique sur le fonctionnement de la mémoire et sur le mode de représentation qu'elle utilise. Nous avons donc dû nous baser sur des hypothèses non démontrées mais toutefois très vraissemblables.

### VI.1.2 La formule d'évaluation

Le caractère imparfait de cette formule est également à souligner. Elle est intéressante car elle tient compte de plusieurs facteurs, dont le temps, mais son échelle n'est pas fondée sur des notions scientifiques. Toutefois, il est aujourd'hui impossible d'imaginer une formule parfaite puisque nous ignorons encore avec certitude tous les facteurs influençant l'oubli.

### VI.1.3 Mémorisation égale association

Nous avons aussi, dès le départ, posé une hypothèse restrictive sur le système. Celui-ci ne peut aider qu'à une mémorisation vue comme une association de représentation de concepts.

Or, nous avons, tous, nos moyens pour retenir plus facilement; ceux-ci dépendent de chaque individu, de son environnement et de sa culture. Ils ne sont donc pas généralisabes.

### VI.1.4 La bonne réponse

Une réponse à une question peut, dans l'absolu, ne pas être tout à fait bonne ou tout à fait fausse.

Nous n'avons pas non plus tenu compte de cette éventualité et une recherche plus approfondie sur le sujet pourrait s'avérer utile.

## VI.2 Utilisation d'une maquette

Faute de temps, nous n'avons pu développer qu'un mini-système d'aide à la mémorisation. Nous avons, cependant, choisi un exemple de maquette qui permette de développer autant que possible, les différents points de ce travail.

## VI.3 Outils d'interface du système

### VI.3.1 Interface élève-système

Cet interface mériterait, à lui seul, des recherches approfondies pour permettre de présenter à l'élève la matière de la façon la plus conviviale possible.

Ceci nous semble très important pour que le mémorisant soit placé dans des circonstances favorisant au mieux son apprentissage.

Il faudrait aussi pouvoir tenir compte un maximum des différents moyens de sensation des élèves.

En effet, on peut imaginer que des personnes aient, par exemple, une meilleure mémoire visuelle qu'auditive.

### VI.3.2 Interface élève-professeur

Ici aussi, il faudrait créer des outils qui puissent aider le plus possible le professeur quand il définit la matière au système.

## VI.4 Le type d'association "se sert de"

Nous sommes conscient du fait que ce type d'association qui lie les types d'entité "fragconn." et "fragmétaconn." n'a pas clairement pu être analysé. Nous avons juste constaté que la relation était many to many.

## VI.5 L'expérimentation

Notons également qu'aucune expérimentation du système n'a pu être faite et que nous ne pouvons donc pas affirmer sa réelle utilité. Une telle analyse pourrait se montrer très intéressante.

# CONCLUSION GENERALE

Nous avons voulu montrer, par ce travail, qu'il était possible de créer des systèmes d'enseignement assisté par ordinateur plus intéressants que les didacticiels existant actuellement.

Par intéressants, nous voulons dire plus généraux et surtout moins fastidieux à l'emploi pour le professeur.

Notre système est plus général puisqu'il permet d'aider à mémoriser toute matière structurée sous forme de relation. Il est moins fastidieux puisqu'il autorise une description synthétique de la métaconnaissance et de la connaissance, quand c'est possible pour cette dernière.

Dans ce travail, nous avons tout d'abord réfléchi à ce qu'était la mémorisation et à ce que signifiait mémoriser (chapitres I et II). Nous avons pu en tirer des conclusions qui ont permis de spécifier le système général dans le chapitre III.

Nous avons, enfin, développé une maquette dans laquelle nous avons pu illustrer les différentes possibilités du système d'une façon simple mais réaliste.Nous avons ainsi prouvé qu'il est possible, à l'avenir, d'envisager des outils informatiques plus adéquats pour l'enseignement.

## CONSIDERATION PERSONNELLE

Au commencement de l'année scolaire 1987-1988, le début de ce travail fut très laborieux. Nous nous attendions, en fait, à pouvoir utiliser différentes méthodologies qui nous auraient mené mécaniquement à la fin.

Nous nous sommes rendu compte, qu'en fait, il n'y avait pas de recette miracle. Il fallait réfléchir au problème d'une manière ordonnée mais il n'existait pas la méthodologie à appliquer.

Ce fut, pour nous, la partie la plus difficile : pouvoir mettre en lumière toutes les notions sous-jacentes au système et bien en maîtriser la teneur.

C'est la première fois que nous devions, à partir de rien, créer un système. C'est pourquoi, cette étape initiale d'analyse s'est révélée, de loin, être la plus importante.

Elle nous a permis, non seulement, de prendre conscience de cette difficulté, mais aussi, de devoir la surmonter.

Au terme de ce travail, nous pourrions formuler le regret suivant: il est dommage que, faute de temps, nous n'ayons pu mener à bien le développement de tout le système. Nous regrettons tout particulièrement le fait de n'avoir pu l'expérimenter avec des élèves.

Mais toutes ces choses étaient trop nombreuses pour être traîtées en un an, et il serait peut-être intéressant pour quelqu'un, qui, comme nous,serait intéressé par le sujet, de prolonger ce travail. Différentes pistes de recherches sont proposées dans le chapitre présentant les limites du système.

# ANNEXE : Texte du programme de la maquette.

```
/*********************************************************/
/*                                                       */
/*            APPLICATION PROFESSEUR                     */
/*                                                       */
/*********************************************************/

domains file = f;g

INCLUDE "database.DCL"
/* insertion de la déclaration de la database */
/* cela permet  de la redefinir dynamiquement */
/* en utilisant cette déclaration comme un fichier */
/* texte */

predicates
     createmetatxt
     suitemetatxt(string)
     trtnivconn(string)
     trteval(string)
     ajregle(string,string)
     ajconnu(string,string)
     ajeval(string,string,string)
     createconntxt
     createtableverif(file)
     verifocc(file,string)
     comp(string,string,integer)
     anal(string,string)
     nbrlettre(string,integer)
     initdate
     initcpt
     finlectocc
     ligneblanche
     verifregle
     existeco(string)
     insertion
     menu
     choix(char)
     ajdeclrel
     ajco
     ajcosynt
      analstrsynt(string,string,integer,integer,integer,integer)
      renvoie(string,string,string,string,string,string,string,string,intege
eger,integer,integer)
     ajmeta
     nouvrel
     nouvregle
     nouvocc
     insnouvrel
     insnouvregle
     maz
     lireregle
     inssynt
     decompsynt(string,integer,integer,integer,integer)
     finsynt(integer,integer)
     dec(string,integer,integer,integer)
     an(string,integer,integer)

goal

    /* APPLICATION 1 */


     makewindow(1,7,1,"PROFESSEUR",0,0,25,80),
     menu,
     clearwindow.


clauses
/*********************************************************************/
/* MENU PROFESSEUR  */
```

```
   menu :- closefile(f),
           closefile(g),
           writedevice(screen),
           readdevice(keyboard),
           shiftwindow(1),
           clearwindow,
      /*   cursor(7,10),
           write("1...Déclaration du type de nouvelles relations\n"),*/
cursor(8,10),
           write("1...Ajout de fragments de connaissance\n"),
           cursor(9,10),
           write("2...Ajout de règles de métaconnaissance\n"),
           cursor(10,10),
           write("3...Ajout de connaissance de façon synthétique.une à la f
"),
           cursor(11,10),
           write("4...Fin de travail\n"),
           cursor(20,10),
           write("Votre choix: "),
           readchar(X),
           choix(X).

   menu :- menu.

   /*choix(X):- X='1',ajdeclrel,menu.*/
   choix(X):- X='1',ajco,menu.
   choix(X):- X='2',ajmeta,menu.
   choix(X):- X='3',ajcosynt,menu.
   choix(X):- X='4',removewindow.


/*********************************************************/
/*                  OUTIL POUR DESSIN                    */
/*********************************************************/
ligneblanche :- write("
                      ").



/*********************************************************/
/*                  OUTILS   DIVERS                      */
/*********************************************************/


/* anal donne le nom d'une occurence de relation */

    anal(Str,Nom):- fronttoken(Str,Nom,_).

/* nbrlettre donne le nombre de lettre du 1er mot d'un
   string avant une parenthese */


    nbrlettre(Str,L) :-  fronttoken(Str,Token,_),
                         str_len(Token,L), write(L).

/* comp regarade si les deux string sont identiques, Rep=1 */

  comp(Str1,Str2,Rep):-Str1=Str2,
                        Rep =1.
  comp(_,_,Rep):-Rep=0.

 maz :- write("0").
/*****************************************************************/
/* outil de vérification de la conn donnéé par le prof */

createtableverif(f):-not(eof(f)),
                     readdevice(f),
                     readln(Str),
                     readdevice(keyboard),
                     anal(Str,Nom),
                     asserta(table(Nom)),
```

```
createtableverif(T).

createtableverif(_).

/*************************************************************/
/*                    MODULE CONNAISSANCE                    */
/*************************************************************/

INCLUDE "conn.pro"

/*************************************************************/
/*                  MODULE METACONNAISSANCE                  */
/*************************************************************/

INCLUDE "metaconn.pro"
```

```
clauses

/*********************************************************/
/*                  MODULE     CONNAISSANCE              */
/*********************************************************/

/* AJDECLREL */


     ajdeclrel:- insnouvrel,
                 readdevice(keyboard),
                 writedevice(screen).

     nouvrel  :- cursor(11,5),
                 write("\nvoulez-vous inserer une nouvelle relation ? "),
                 readchar(X),
                 cursor(12,1),
                 ligneblanche,
                 X='o'.


     insnouvrel  :-clearwindow,
                   cursor(11,2),
                   write("\nVeuillez entrer la déclaration de relation: "),
                   readln(Declrel),
                   cursor(12,1),
                   ligneblanche,
                   not(existeco(Declrel)),
                   openmodify(f,"database.dcl"),
                   writedevice(f),
                   readdevice(f),
                   readln(_),
                   readln(_),
                   readln(_),
                   str_len(Declrel,L),
                   J=L-1,
                   frontstr(J,Declrel,P1,_),
                                        /* COTE  ANNEE   MOIS     JOUR   H
  MINUTE SECONDE DIXIEME NbQUES  NbBonnrep SELECTION?*/
                   fronttoken(Declcomp,P1,",real,integer,integer,integer,ir
,integer,integer,integer,integer,integer,integer)\n"),
                   write(Declcomp),
                   closefile(f),
                   openmodify(f,"metaconn.dcl"),
                   writedevice(f),
                   readdevice(f),
                   readln(_),
                   anal(Declrel,Pn),
                   fronttoken(Mot,Pn,"connu(integer)\n"),
                   write(Mot),
                   closefile(f),
                   writedevice(screen),
                   readdevice(keyboard).

     insnouvrel:- clearwindow,
                  cursor(6,1),
                  write("erreur ce nom est impossible\n").


 /*AJCONN */
     ajco :- insertion.
     nouvocc  :- cursor(11,2),
                 write("\nVoulez-vous insérer une nouvelle occurence de rel
 ? "),
                 readchar(X),
                 cursor(12,1),
                 ligneblanche,
                 X='o'.

     insertion:-  /* nouvelle occurence de relation */
```

```
                        clearwindow,
                        cursor(11,2),
                        write("La nouvelle occurence de relation: "),
                        readln(Occrel),
                        existeco(Occrel),
                        openmodify(f,"conn     .txt"),
                        writedevice(f),
                        readdevice(f),
                        filepos(f,0,2),
                        str_len(Occrel,L),
                        J=L-1,
                        frontstr(J,Occrel,P1,_),
                        write(P1,","),
                        maz,
                        initdate,
                        initcpt,
                        closefile(f),
                        writedevice(screen),
                        readdevice(keyboard).


    finlectocc:-cursor(11,2),
                write("\nvoulez-vous inserer une nouvelle occurence de rel
"),
                readchar(X),
                cursor(12,1),
                ligneblanche,
                X='o'.

    initdate:- date(An,Mo,Jo),
               time(He,Mi,Se,Di),
               write(",",An,",",Mo,",",Jo,",",He,",",Mi,",",Se,",",Di).

    /* compteur de bonne réponse et du nombre de fois posé et de selectior
*/

    initcpt :-write(",0,0,0)","\n").

    /* SELCONN dans le fichier conn.txt */


/*AJCONN SYNTHETIQUE */
    ajcosynt :-     openwrite(f,"conn.ext"),
                    writedevice(f),inssynt,
                    writedevice(screen),closefile(f),
                    openread(g,"conn.ext"),
                    readdevice(g),createconntxt,
                    readdevice(keyboard),closefile(g).

    inssynt  :-  /* nouvelle description syntétique de relation */

                    clearwindow,
                    cursor(11,2),
                    writedevice(screen),
                    write("La nouvelle description de relation: "),
                    writedevice(f),
                    readln(Occrel),
                    str_len(Occrel,L),
                    L>0,
                    write(Occrel,"\n").

    inssynt.

    createconntxt :-readln(Desc),
                    analstrsynt(Desc,Str,X1,Y1,Xn,Yn),
                    existeco(Str),
                    decompsynt(Str,X1,Y1,Xn,Yn).
    createconntxt.
```

```
analstrsynt(Desc,Ndrel,X1,Y1,Xn,Yn):-
                    fronttoken(Desc,Ndrel,Reste1),
                    frontstr(10,Reste1,_,Reste2),
                    fronttoken(Reste2,_,Reste3),
                    frontstr(2,Reste3,Premin,Reste4),
                    frontstr(1,Reste4,_,Reste5),
                    fronttoken(Reste5,_,Reste6),
                    frontstr(2,Reste6,Deuxin,Reste7),
                    frontstr(1,Reste7,_,Reste8),
                    str_len(Reste8,L),
                    L=0,
                    str_len(Premin,L1),
                    L2=L1-1,
                    frontstr(L2,Premin,_,A1str),
                    str_len(Premin,L3),
                    L4=L3-1,
                    frontstr(L4,Deuxin,_,A2str),
                    /* je n'ai que deux inégalité. ca ne peut être que les
                    /* inutile dans ce cas renvoie(Signe1,Signe2,_,_,X1str
r,_,_,X1,X2,_,_).*/
                    str_int(A1str,Xn),
                    str_int(A2str,Yn),
                    X1=0,Y1=0.

analstrsynt(Desc,Ndrel,X1,Y1,Xn,Yn):-
                    fronttoken(Desc,Ndrel,Reste1),
                    frontstr(10,Reste1,_,Reste2),
                    fronttoken(Reste2,_,Reste3),
                    frontstr(2,Reste3,Premin,Reste4),
                    frontstr(1,Reste4,_,Reste5),
                    fronttoken(Reste5,_,Reste6),
                    frontstr(2,Reste6,Deuxin,Reste7),
                    frontstr(1,Reste7,_,Reste8),
                    fronttoken(Reste8,_,Reste9),
                    frontstr(2,Reste9,Troisin,Reste10),
                    frontstr(1,Reste10,_,Reste11),
                    fronttoken(Reste11,_,Reste12),
                    frontstr(2,Reste12,Quatin,_),
                    str_len(Premin,L1),
                    str_len(Deuxin,L2),
                    str_len(Troisin,L3),
                    str_len(Quatin,L4),
                    L1b=L1-1,
                    L2b=L2-1,
                    L3b=L3-1,
                    L4b=L4-1,
                    frontstr(L1b,Premin,Signe1,A1str),
                    frontstr(L2b,Deuxin,Signe2,A2str),
                    frontstr(L3b,Troisin,Signe3,A3str),
                    frontstr(L4b,Quatin,Signe4,A4str),
                    renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,
,X1,Y1,Xn,Yn).

renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                    Signe1 ="<",
                    Signe2= "<",
                    Signe3= ">",
                    Signe4= ">",
                    str_int(A1str,Xn),
                    str_int(A2str,Yn),
                    str_int(A3str,X1),
                    str_int(A4str,Y1).

renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                    Signe1 ="<",
                    Signe2= ">",
                    Signe3= "<",
                    Signe4= ">",
                    str_int(A1str,Xn),
                    str_int(A2str,X1),
                    str_int(A3str,Yn),
```

```
renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                Signe1 ="<",
                Signe2= ">",
                Signe3= ">",
                Signe4= "<",
                str_int(A1str,Xn),
                str_int(A2str,X1),
                str_int(A3str,Y1),
                str_int(A4str,Yn).

renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                Signe1 =">",
                Signe2= ">",
                Signe3= "<",
                Signe4= "<",
                str_int(A1str,X1),
                str_int(A2str,Y1),
                str_int(A3str,Xn),
                str_int(A4str,Yn).

renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                Signe1 =">",
                Signe2= "<",
                Signe3= ">",
                Signe4= "<",
                str_int(A1str,X1),
                str_int(A2str,Xn),
                str_int(A3str,Y1),
                str_int(A4str,Yn).

renvoie(Signe1,Signe2,Signe3,Signe4,A1str,A2str,A3str,A4str,X1,Y1,Xn,Y
                Signe1 =">",
                Signe2= "<",
                Signe3= "<",
                Signe4= ">",
                str_int(A1str,X1),
                str_int(A2str,Xn),
                str_int(A3str,Y1),
                str_int(A4str,Yn).

decompsynt(Declsynt,X1,Y1,Xn,Yn) :-
                not(finsynt(X1,Xn)),
                dec(Declsynt,X1,Y1,Yn),
                NouvX1 = X1+1,
                decompsynt(Declsynt,NouvX1,Y1,Xn,Yn).
decompsynt(_,_,_,_,_,_).

dec(Declsynt,X1,Y1,Yn) :-
                not(finsynt(Y1,Yn)),
                frontstr(1,Declsynt,Premlet,_),
                an(Premlet,X1,Y1),
                NouvY1 = Y1+1,
                dec(Declsynt,X1,NouvY1,Yn).

 dec(_,_,_,_).
 an(Lettre,X1,Y1) :- Lettre ="p",
                openmodify(f,"conn     .txt"),
                writedevice(f),
                readdevice(f),
                filepos(f,0,2),
                write("plus("),
                Z=X1+Y1,
                write(X1,",",Y1,",",Z,","),
                maz,
                initdate,
                initcpt,
                closefile(f),
                writedevice(screen),
                readdevice(keyboard).
```

```
an(Lettre,X1,Y1) :- Lettre = Y1,
                    openmodify(f,"conn     .txt"),
                    writedevice(f),
                    readdevice(f),
                    filepos(f,0,2),
                    write("fois("),
                    Z=X1*Y1,
                    write(X1,",",Y1,",",Z,","),
                    maz,
                    initdate,
                    initcpt,
                    closefile(f),
                    writedevice(screen),
                    readdevice(keyboard).
     an(_,_,_).

     finsynt(A,B):- A>=B.


/* SELCO */
/* se fait automatiquement en appelant le nom de relation */


/* EXISTECO */
     existeco(Occrel):-      openread(f,"database.dcl"),
                             readdevice(f),
                             readln(_),/* lecture du mot database*/
                             readln(_),/* lecture de la declaration de tabl
                             readdevice(keyboard),
                             verifocc(f,Occrel),
                             closefile(f).


     verifocc(f,Str):- createtableverif(f),
                       table(X),
                       anal(Str,Nom),
                       comp(X,Nom,Rep),
                       Rep=1.

     verifocc(f,_):-closefile(f),fail.


/* SUPCO */
/* utiliser le retract de prolog */
```

```
clauses
/*********************************************************/
/*                MODULE       METACONNAISSANCE          */
/*********************************************************/

/* AJMETACONN */

     ajmeta:-     shiftwindow(1),
                  clearwindow,
                  makewindow(2,7,1,"",22,10,3,60),
                  makewindow(3,7,7,"Regle",11,2,3,77),
                  insnouvregle,
                  shiftwindow(1),
                  readdevice(keyboard),
                  writedevice(screen).

     nouvregle:- cursor(11,2),
                  write("\nvoulez-vous inserer une nouvelle regle? "),
                  readchar(X),
                  ligneblanche,
                  X='o'.
     insnouvregle :-  clearwindow,
                      openwrite(f,"metaconn.ext"),

                      /* Déclaration d'une nouvelle règle */
                      lireregle,
                      closefile(f),
                      writedevice(screen),
                      openread(f,"metaconn.ext"),
                      openmodify(g,"metaconn.txt"),
                      readdevice(f),
                      writedevice(g),
                      createmetatxt,
                      writedevice(screen),
                      readdevice(keyboard),
                      closefile(g),
                      closefile(f).

     lireregle :-
                    shiftwindow(2),
                    write("                    taper return pour arrêter"),
                    shiftwindow(3),
                    readln(Regle),
                    str_len(Regle,L),
                    L>0,
                    cursor(11,2),
                    ligneblanche,
                    verifregle,
                    filepos(f,0,2),
                    writedevice(f),
                    write(Regle,"\n"),
                    writedevice(screen),
                    lireregle.
     lireregle.

     verifregle./* pour n'insérer que des règles bien formées*/

/* AJMETACONN */


createmetatxt :-     filepos(g,0,2),
                     readdevice(f),
                     readln(Regle),
                     frontstr(3,Regle,Type,Reste1),
                     frontstr(2,Reste1,_,Rel),
                     writedevice(g),
                     aireale(Rel,Type),
```

```
                        not(eof(f)),createmetatxt.

createmetatxt :-        filepos(g,0,2),
                        filepos(f,0,0),
                        readdevice(f),
                        readln(Regle),
                        suitemetatxt(Regle).

suitemetatxt(Regle) :-trtnivconn(Regle).
suitemetatxt(Regle) :-trteval(Regle).
suitemetatxt(_).

trtnivconn(Regle) :- /* pour traiter les xCONNU */
                        fronttoken(Regle,Ndrel,Reste1),
                        str_len(Ndrel,Lr),
                        Lr2=Lr-5,
                        frontstr(Lr2,Ndrel,_,Test),
                        Test="connu",
                        str_len(Reste1,L),
                        K=L-1,
                        frontstr(K,Reste1,Reste2,_),
                        J=K-1,
                        frontstr(J,Reste2,_,Cote),
                        writedevice(g),
                        ajconnu(Ndrel,Cote),
                        readdevice(f),
                        readln(R),
                        suitemetatxt(R).


trteval(Regle)          :- /* pour traiter les eval */
                        frontstr(5,Regle,Test,Reste1),
                        Test="eval(",
                        frontstr(1,Reste1,Var1,Reste2),
                        frontstr(1,Reste2,_,Reste3),
                        frontstr(1,Reste3,Var2,Reste4),
                        frontstr(3,Reste4,_,Reste5),
                        str_len(Reste5,L),
                        K=L-1,
                        frontstr(K,Reste5,Txt,_),
                        writedevice(g),
                        ajeval(Var1,Var2,Txt),
                        not(eof(f)),
                        readdevice(f),
                        readln(R),
                        suitemetatxt(R).


/* pour ajouter les règles eval à metaconn.txt*/
ajeval(Var1,Var2,Txt) :- write("eval(",Var1,",",Var2,"):-"),
                         write(Txt),
                         write(",readchar(_),clearwindow,shiftwindow(2).",'
\n").

ajeval(_,_,_).


/* pour ajouter les règles plusconnu à metaconn.txt). */
ajconnu(Rel,Cote):- write(Rel,":-"),str_len(Rel,L),
                    K=L-5,
                    frontstr(K,Rel,Drel,_),
                    write(Drel,"(_,_,_,Cote,_,_,_,_,_,_,_,_,_,_),Cote>",Co
,"\n","\n").

ajconnu(_,_).
```

```
/* pour ajouter les régles iad,qap,iaa,qaa à metaconn.txt */
                /*************************/
                /* Question à poser      */
                /*************************/


ajregle(R,Type):-

            Type="qap",
            fronttoken(R,Par1,Reste1),
            frontstr(7,Reste1,Par2,Reste2),
            fronttoken(Rel,Par1,Par2),
            frontstr(4,Rel,Ndrel,_),
            str_len(Rel,L),
            K=L-1, /* enlever parenthese fermante*/
            frontstr(K,Rel,Deb,_),
            write("qap:-"),
            write(Deb,",Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1),",","\n"),
            write("Cote<=10,","\n"),
            frontstr(1,Reste2,_,Suite),
            str_len(Suite,J),
            I=J-1, /* enlever le point */
            frontstr(I,Suite,Nsuite,_),
            write(Nsuite,",\n"),
            write("lireentier(Rep),","\n"),
            write("majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),","\n"),
            write("Ndrel ="),
            write("\34"),
            write(Ndrel),
            write("("),
            write("\34"),
            write(",\n"),
            write("transfhist(Ndrel,X,Y,Z,Question),","\n"),
            write("str_int(Le,Rep),","\n"),
            write("ajhis(Question,Le),","\n"),
            write("date(Aa,Mm,Jj),","\n"),
            write("time(Hh,Mn,Ss,Dd),","\n"),
            write("N=N1+1,","\n"),
            write("retract("),
            write(Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)),","\n"),
            write("assert("),
            write(Ndrel,"(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,0)),","\n"),
            write("not(pause),","\n"),
            write("shiftwindow(2).","\n","\n").

                /*************************/
                /* Informations à donner*/
                /*************************/

ajregle(R,Type)
        :- Type="iad",
            fronttoken(R,Par1,Reste1),
            frontstr(7,Reste1,Par2,Reste2),
            fronttoken(Rel,Par1,Par2),
            str_len(Rel,L),
            K=L-1, /* enlever parenthese fermante*/
            frontstr(K,Rel,Deb,_),
            frontstr(4,Deb,Ndrel,_),
            write("iad:-"),
            write(Deb,",Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0),",","\n"),
            write("Cote<=10,","\n"),
            write("failcount(I),","\n"),
            write("J=I+1,","\n"),
            write("I<=7,"),
            frontstr(1,Reste2,_,Suite),
            str_len(Suite,J),
            I=J-1, /* enlever le point */
            frontstr(I,Suite,Nsuite,_),
            write(Nsuite),
```

```
                write(",nl,"),
                write("\n"),
                write("asserta(failcount(J)),","\n"),
                write("retract(failcount(I)),","\n"),
                write("retract("),
                write(Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0)),","\n"),
                write("assertz("),
                write(Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)).","\n","

                        /*************************/
                        /*ANALYSE DE QAA         */
                        /*************************/

ajregle(Regle,Type) :- Type="qaa",
                       write("anal1(Inf,Z):-  fronttoken(Inf,Ndrel,Reste1),

"),
                       fronttoken(Regle,Ndrel,R1),
                       frontstr(8,R1,_,R2),
                       str_len(R2,L),
                       K=L-1,
                       frontstr(K,R2,Reste,_),
                       write("Ndrel="),write("\34"),
                       write(Ndrel,"\34,","\n"),
                       write("frontstr(1,Reste1,_,Reste1bis),","\n"),
                       write("frontstr(1,Reste1bis,Xstr,Reste2),","\n"),
                       write("str_int(Xstr,X),","\n"),
                       write("frontstr(1,Reste2,_,Reste3),","\n"),
                       write("frontstr(1,Reste3,Ystr,_),","\n"),
                       write("str_int(Ystr,Y),","\n"),
                       write(Reste,",","\n"),
                       write(Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,

"\n"),
                       write("!,"),
                       write("Rep =Z+1,","\n"),
                       write("majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,N

"\n"),
                       write("date(Aa,Mm,Jj),","\n"),
                       write("time(Hh,Mn,Ss,Dd),","\n"),
                       write("N=N1+1,","\n"),
                       write("retract(",Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,S

N1,N2,Tt)),","\n"),
                       write("assert(",Ndrel,"(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,D

n2,Tt)).","\n","\n").


                        /*************************/
                        /*ANALYSE DE IAA         */
                        /*************************/

ajregle(Regle,Type) :- Type="iaa",
                       write("anal2(Inf):-  fronttoken(Inf,Ndrel,Reste1),",

,
                       fronttoken(Regle,Ndrel,R1),
                       frontstr(8,R1,_,R2),
                       str_len(R2,L),
                       K=L-1,
                       frontstr(K,R2,Reste,_),
                       write("Ndrel="),write("\34"),
                       write(Ndrel,"\34,","\n"),
                       write("frontstr(1,Reste1,_,Reste1bis),","\n"),
                       write("frontstr(1,Reste1bis,Xstr,Reste2),","\n"),
                       write("str_int(Xstr,X),","\n"),
                       write("frontstr(1,Reste2,_,Reste3),","\n"),
                       write("frontstr(1,Reste3,Ystr,Reste4),","\n"),
                       write("str_int(Ystr,Y),","\n"),
                       write("frontstr(1,Reste4,_,Reste5),","\n"),
                       write("frontstr(1,Reste5,Zstr,_),","\n"),
                       write("str_int(Zstr,Z),","\n"),
                       write(Reste,",","\n"),
                       write(Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2
```

```
"\n"),
                                write("!,"),
                                write("Rep =Z+1,","\n"),
                                write("majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,N
"\n"),
                                write("date(Aa,Mm,Jj),","\n"),
                                write("time(Hh,Mn,Ss,Dd),","\n"),
                                write("N=N1+1,","\n"),
                                write("retract(",Ndrel,"(X,Y,Z,Cote,An,Mo,Jo,He,Mi,S
N1,N2,Tt)),","\n"),
                                write("assert(",Ndrel,"(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,D
n2,Tt)).","\n","\n").

ajregle(_,_).
```

```
/*************************************************************/
/*                                                           */
/* UN EXEMPLE DE TEXTE DE LA METACONNAISSANCE: REGLES        */
/*   REPRESENTATION EXECUTABLE                               */
/*************************************************************/

clauses

                /*************************/
                /*  Questions à poser    */
                /*************************/


qap :-
     fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1),Cote<3,
     write(X," fois ",Y," = ").
```

```
      lireentier(Rep),
      majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
      Ndrel ="fois(",
      transfhist(Ndrel,X,Y,Z,Question),
      str_int(Le,Rep),
      ajhis(Question,Le),
      date(Aa,Mm,Jj),
      time(Hh,Mn,Ss,Dd),
      N=N1+1,
      retract(fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)),
      assert(fois(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,0)),
      not(pause),
      shiftwindow(2).

qap:-
      plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1),
      Cote<8,
      write(X," plus ",Y," = "),
      lireentier(Rep),
      majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
      Ndrel ="plus(",
      transfhist(Ndrel,X,Y,Z,Question),
      str_int(Le,Rep),
      ajhis(Question,Le),
      date(Aa,Mm,Jj),
      time(Hh,Mn,Ss,Dd),
      N=N1+1,
      retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)),
      assert(plus(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,0)),
      not(pause),
      shiftwindow(2).


                /*************************/
                /* Informations à donner*/
                /*************************/


iad:-   plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0),
        Cote<8,
        failcount(I),
        J=I+1,
        I<=7,write(X," plus ",Y," = ",Z),nl,
        asserta(failcount(J)),
        retract(failcount(I)),
        retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0)),
        assertz(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)).


iad:-  plusconnu,
       fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0),
       Cote<8,
       failcount(I),
       J=I+1,
       I<=7,
       write(X," fois ",Y," = ",Z),nl,
       asserta(failcount(J)),
       retract(failcount(I)),
       retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,0)),
       assertz(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,1)).

plusconnu:-plus(_,_,_,Cote,_,_,_,_,_,_,_,_,_),Cote>8.

foisconnu:-fois(_,_,_,Cote,_,_,_,_,_,_,_,_,_),Cote>8.

eval(X,Y):-X<>Y,write("C'est ",X),readchar(_),clearwindow,shiftwindow(2).

eval(X,Y):-X=Y,write("c'est bon"),readchar(_),clearwindow,shiftwindow(2).

anal1(Inf,Z):-  fronttoken(Inf,Ndrel,Reste1),
Ndrel="plus",
```

```
frontstr(1,Reste1,_,Reste1bis),
frontstr(1,Reste1bis,Xstr,Reste2),
str_int(Xstr,X),
frontstr(1,Reste2,_,Reste3),
frontstr(1,Reste3,Ystr,_),
str_int(Ystr,Y),
X<2,Y<2,
plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,_),
!,Rep =Z+1,
majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
date(Aa,Mm,Jj),
time(Hh,Mn,Ss,Dd),
N=N1+1,
retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,Tt)),
assert(plus(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,Tt)).

anal1(Inf,Z):-  fronttoken(Inf,Ndrel,Reste1),
Ndrel="fois",
frontstr(1,Reste1,_,Reste1bis),
frontstr(1,Reste1bis,Xstr,Reste2),
str_int(Xstr,X),
frontstr(1,Reste2,_,Reste3),
frontstr(1,Reste3,Ystr,_),
str_int(Ystr,Y),
X<2,Y<2,
fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,_),
!,Rep =Z+1,
majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
date(Aa,Mm,Jj),
time(Hh,Mn,Ss,Dd),
N=N1+1,
retract(fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,Tt)),
assert(fois(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,Tt)).

anal2(Inf):-  fronttoken(Inf,Ndrel,Reste1),
Ndrel="plus",
frontstr(1,Reste1,_,Reste1bis),
frontstr(1,Reste1bis,Xstr,Reste2),
str_int(Xstr,X),
frontstr(1,Reste2,_,Reste3),
frontstr(1,Reste3,Ystr,Reste4),
str_int(Ystr,Y),
frontstr(1,Reste4,_,Reste5),
frontstr(1,Reste5,Zstr,_),
str_int(Zstr,Z),
X<2,Y<2,
plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,_),
!,Rep =Z+1,
majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
date(Aa,Mm,Jj),
time(Hh,Mn,Ss,Dd),
N=N1+1,
retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,Tt)),
assert(plus(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,Tt)).

anal2(Inf):-  fronttoken(Inf,Ndrel,Reste1),
Ndrel="fois",
frontstr(1,Reste1,_,Reste1bis),
frontstr(1,Reste1bis,Xstr,Reste2),
str_int(Xstr,X),
frontstr(1,Reste2,_,Reste3),
frontstr(1,Reste3,Ystr,Reste4),
str_int(Ystr,Y),
frontstr(1,Reste4,_,Reste5),
frontstr(1,Reste5,Zstr,_),
str_int(Zstr,Z),
X<2,Y<2,
fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,_),
!,Rep =Z+1,
majcote(Rep,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,Nn2,Nc),
date(Aa,Mm,Jj),
time(Hh,Mn,Ss,Dd),
```

```
N=N1+1,
retract(fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,Tt)),
assert(fois(X,Y,Z,Nc,Aa,Mm,Jj,Hh,Mn,Ss,Dd,N,Nn2,Tt)).
```

```
/*************************************************************/
/*                                                           */
/*              APPLICATION       ELEVE                      */
/*                                                           */
/*************************************************************/

code=1536

domains file = f ;g

INCLUDE "database.DCL"
/* insertion de la déclaration de la database          */
/* cela permets de la redefinir dynamiquement          */
/* en utilisant cette déclaration comme un fichier     */
/* texte                                               */


predicates
     initconn
     lire(integer)
     moteur
     maz
     iadonner
     qaposer
     testdollar(string)
     stop
     term
     majcote(integer,integer,real,integer,integer,integer,integer,integer,in
,integer,integer,real)
     eval(integer,integer)
     iad
     pause
     qap
     anal1(string,integer)
     anal2(string)
     fin
     selcote(integer,integer,real)
     calcstat(integer,integer,real)
     ajhis(string,string)
     transfhist(string,integer,integer,integer,string)
     inshist(file,file,string,string)
     declheure
     lireentier(integer)
     analentier(string,integer)


 INCLUDE "metaconn.dcl"


goal

     /* APPLICATION 2 : L'ELEVE MEMORISE */

     declheure,
     consult ("conn.txt"),
     initconn,
     makewindow(1,7,1,"TABLES D'ADDITIONS ET DE MULTIPLICATIONS",0,0,25,80),
     makewindow(4,7,7,"COMMENTAIRES",2,33,5,42),
     makewindow(5,7,1,"",22,10,3,60),
     makewindow(3,7,7,"INFORMATIONS",2,2,10,30),
     makewindow(2,7,7,"QUESTIONS",14,20,7,34),
     makewindow(7,7,7,"VOTRE QUESTION",7,33,3,42),
     makewindow(6,7,7,"VOTRE INFORMATION",11,33,3,42),
     moteur.


clauses
/*************************************************************/
/*                  MODULE       MOTEUR                      */
/*************************************************************/
```

```
/* lire une question que pose l'élève */
moteur:- shiftwindow(7),
         readln(Inf),
         str_len(Inf,L),
         testdollar(Inf),
         L>0,
         anal1(Inf,_),
         clearwindow,fail.


/* lire une information que donne l'élève*/
moteur:-  shiftwindow(6),
         readln(Inf),str_len(Inf,L),
         testdollar(Inf),
         L>0,
         anal2(Inf),
         clearwindow,fail.


/* sélectionner une information à donner */
moteur:- shiftwindow(3),
         iadonner,
         clearwindow.


/* sélectionner une question à poser */
moteur:- shiftwindow(5),
         write("          taper $ pour reprendre le controle"),
         shiftwindow(2),
         qaposer,
         clearwindow.

moteur:- moteur.


/*********************************************************/
/* sélection d'une information à donner */

iadonner :-   shiftwindow(3),pause,
                  iad,
                 fail.

iadonner :-    retract(failcount(_)),
               asserta(failcount(1)),
               !,
               shiftwindow(5),
               write( "      appuyer sur une touche quand vous avez bien lu
               readchar(_),clearwindow,
               shiftwindow(3),
               clearwindow,fail.



/* sélectionner une question à poser */

    qaposer:-  shiftwindow(2),clearwindow,
               qap,shiftwindow(5),clearwindow,
               shiftwindow(2),fail.



/* Stopper le moteur */
    testdollar(Inf):- str_len(Inf,L),
                      L=1,
                      str_char(Inf,C),
                      C='$',
                      not(stop).
    testdollar(_).
```

```
        shiftwindow(5),
        write("                          Voulez-vous arrêter ?"),
        readchar(X),clearwindow,
        X='o',
        retract(failcount(_)),
        asserta(failcount(1)),
        save("conn.txt"),
        removewindow,
        shiftwindow(4),removewindow,
        shiftwindow(1),removewindow,
        shiftwindow(3),removewindow,
        shiftwindow(2),removewindow,
        shiftwindow(7),removewindow,
        shiftwindow(6),removewindow.


/* Outil de lecture 36 = '$'*/

lireentier(Rep) :- readln(Reponse),
                   analentier(Reponse,Rep).

analentier(Reponse,Rep) :-
                   str_len(Reponse,L),
                   L>1,
                   str_int(Reponse,Reponseint),
                   Rep=Reponseint.

analentier(Reponse,Rep) :- /* reponse en un char */
                   str_char(Reponse,Repchar),
                   char_int(Repchar,Repasc),
                   lire(Repasc),
                   Rep = Repasc-48.

    lire(36) :- not(stop),moteur.
    lire(36) :- exit.
    lire(_).


/* déclencher heure de début de mem*/
    declheure :- time(X,Y,_,_),
                 openwrite(f,"hdeb"),
                 writedevice(f),
                 write(X,"\n"),
                 write(Y,"\n"),
                 closefile(f),
                 writedevice(screen).

/* terminaison : remettre à 0 tout les codes de sélection à 1 */

   term :- plus(A,B,C,D,E,F,G,H,I,J,K,L,M,1),
           asserta(plus(A,B,C,D,E,F,G,H,I,J,K,L,M,0)),
           retract(plus(A,B,C,D,E,F,G,H,I,J,K,L,M,1)),
           fail.

   term :- fois(A,B,C,D,E,F,G,H,I,J,K,L,M,1),
           asserta(fois(A,B,C,D,E,F,G,H,I,J,K,L,M,0)),
           retract(fois(A,B,C,D,E,F,G,H,I,J,K,L,M,1)),
           fail.
   term.


/*********************************************************/
pause :- time(X,Y,_,_),
         openread(f,"Hdeb"),
         readdevice(f),
         readint(X1),
         readint(X2),
         closefile(f),
         readdevice(keyboard),
         Tps = abs((X*60+Y)-(X1*60+X2)),
         Tps > 40,
```

```
            shiftwindow(5),
            write("          Il est temps de faire une petite pause"),
            readchar(_),
            write("                          Appuyer sur une touche pour continuer")
            readchar(_),
            clearwindow,
            declheure.

pause.


                    /*************************/
                    /* COND DE FIN DE MEMOR.*/
                    /*************************/


fin:-plusconnu,
     foisconnu.


/*************************************************************/
/*                  MODULE    COTATION                       */
/*************************************************************/

INCLUDE "cotation.pro"

/*************************************************************/
/*                  MODULE   HISTORIQUE                      */
/*************************************************************/

INCLUDE "histori.pro"

/*************************************************************/
/*                  MODULE   METACONNAISSANCE                */
/*************************************************************/

INCLUDE "metaconn.txt"/* uniquement la connaissance,  */
                      /* pas les procedures           */




/*************************************************************/
/*                  MODULE   CONNAISSANCE                    */
/*************************************************************/

/* la connaissance se trouve dans la database               */
```

```
clauses

/*********************************************************/
/*                  MODULE          COTATION             */
/*********************************************************/
majcote(T,Z,Cote,An,Mo,Jo,He,Mi,N1,N2,NouvN2,Nouvcote):-
                  shiftwindow(4),
                  eval(Z,T),
                  Z=T,
                  /* la réponse est bonne */
                  date(A,B,C),time(D,E,_,_),
                  Nbjour = (A-An)*365 + (abs(B-Mo))*31 + abs(C-Jo),
                  Nbmin  = (abs(D-He))*60 + abs(E-Mi),


                  /* modification de la cote % temps */
                  selcote(Nbjour,Nbmin,Mu),!,
                  Ncote = (Cote +4)*Mu,

                  /* modification de la cote % stat */
                  calcstat(N1,N2,St),
                  Nouvcote=Ncote*St,
                  NouvN2 = N2+1.

majcote(_,_,Cote,An,Mo,Jo,He,Mi,N1,N2,NouvN2,Nouvcote):-

                  /* la réponse est fausse */
                  date(A,B,C),time(D,E,_,_),
                  Nbjour = (A-An)*365 + (abs(B-Mo))*31 + abs(C-Jo),
                  Nbmin  = (abs(D-He))*60 + abs(E-Mi),

                  /* modification de la cote % temps */
                  selcote(Nbjour,Nbmin,Mu),!,
                  Ncote= (Cote-4)*Mu,

                  /* modification de la cote % stat */
                  calcstat(N1,N2,St),
                  Nouvcote = Ncote*St,
                  NouvN2=N2.

/*********************************************************/
/* Echelle des points à retirer selon le stat            */

calcstat(N1,N2,St):-N2<>0,N1<>0,St=N1/N2.
calcstat(_,_,St):-St=1.


/*********************************************************/
/* Echelle des points à retirer selon le temps           */


selcote(Nbjour,_,Mu):-
                Nbjour >365,
                Mu=0.2.

selcote(Nbjour,_,Mu):-
                Nbjour>31,
                Nbjour<=365,
                Mu=0.5.

selcote(Nbjour,_,Mu):-
                Nbjour>=1,
                Nbjour<=31,
                Mu=0.7.

selcote(_,Nbmin,Mu):-
                Nbmin>20,
                Mu=0.85.

selcote(_,Nbmin,Mu):-
```

```
                Nbmin<=20,
                Mu=0.90.

selcote(_,Nbmin,Mu):-
                Nbmin>5,
                Nbmin<=10,
                Mu=0.97.

selcote(_,_,Mu):- Mu =0.99.

/* Mise à zero */
maz :- write("0").


/* pour diminuer la cote au début, selon le temps */

initconn:- plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O),
          date(A,B,C),time(D,E,_,_),
          Nbjour = (A-An)*365 + (abs(B-Mo))*31 + abs(C-Jo),
          Nbmin  = (abs(D-He))*60 + abs(E-Mi),
          /* modification de la cote % temps */
          selcote(Nbjour,Nbmin,Mu),
          Ncote = Cote*Mu,
          assert(plus(X,Y,Z,Ncote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O)),
          retract(plus(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O)),
          fail.

initconn:- fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O),
          date(A,B,C),time(D,E,_,_),
          Nbjour = (A-An)*365 + (abs(B-Mo))*31 + abs(C-Jo),
          Nbmin  = (abs(D-He))*60 + abs(E-Mi),
          /* modification de la cote % temps */
          selcote(Nbjour,Nbmin,Mu),
          Ncote = Cote*Mu,
          assert(fois(X,Y,Z,Ncote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O)),
          retract(fois(X,Y,Z,Cote,An,Mo,Jo,He,Mi,Se,Di,N1,N2,O)).
```

```
clauses
/*************************************************************/
/*          MODULE          HISTORIQUE                       */
/*************************************************************/
ajhis(Question,Reponse):- openread(f,"Hist.txt"),
                          openwrite(g,"prov.txt"),
                          writedevice(g),
                          readdevice(f),
                          inshist(f,g,Question,Reponse),!.



/*************************************************************/

 /* insertion du nouvel element au début  */

inshist(f,g,Question,Reponse):-   filepos(g,0,0),
                          write(Question,"\n"),
                          write(Reponse,"\n"),
                          date(An,Mo,Jo),
                          time(He,Mi,_,_),
                          write(An,"\n"),
                          write(Mo,"\n"),
                          write(Jo,"\n"),
                          write(He,"\n"),
                          write(Mi,"\n"),

                          not(eof(f)),
                          readln(Q1),write(Q1,"\n"),
                          readln(R1),write(R1,"\n"),
                          readln(An1),write(An1,"\n"),
                          readln(Mo1),write(Mo1,"\n"),
                          readln(Jo1),write(Jo1,"\n"),
                          readln(He1),write(He1,"\n"),
                          readln(Mi1),write(Mi1,"\n"),
                          not(eof(f)),
                          readln(Q2),write(Q2,"\n"),
                          readln(R2),write(R2,"\n"),
                          readln(An2),write(An2,"\n"),
                          readln(Mo2),write(Mo2,"\n"),
                          readln(Jo2),write(Jo2,"\n"),
                          readln(He2),write(He2,"\n"),
                          readln(Mi2),write(Mi2,"\n"),
                          not(eof(f)),
                          readln(Q3),write(Q3,"\n"),
                          readln(R3),write(R3,"\n"),
                          readln(An3),write(An3,"\n"),
                          readln(Mo3),write(Mo3,"\n"),
                          readln(Jo3),write(Jo3,"\n"),
                          readln(He3),write(He3,"\n"),
                          readln(Mi3),write(Mi3,"\n"),
                          not(eof(f)),
                          readln(Q4),write(Q4,"\n"),
                          readln(R4),write(R4,"\n"),
                          readln(An4),write(An4,"\n"),
                          readln(Mo4),write(Mo4,"\n"),
                          readln(Jo4),write(Jo4,"\n"),
                          readln(He4),write(He4,"\n"),
                          readln(Mi4),write(Mi4,"\n"),
                          fail.

inshist(f,g,_,_):-        closefile(f),
                          closefile(g),
                          renamefile("prov.txt","hist.txt"),
                          writedevice(screen),
                          readdevice(keyboard).


transfhist(Ndrel,X,Y,Z,Question) :-
```

```
Xascii=X+48,
Yascii=Y+48,
Zascii=Z+48,
char_int(Xlprov,Xascii),str_char(Xl,Xlprov),
char_int(Ylprov,Yascii),str_char(Yl,Ylprov),
char_int(Zlprov,Zascii),str_char(Zl,Zlprov),
fronttoken(Qa,Ndrel,Xl),
fronttoken(Qb,Qa,","),
fronttoken(Qc,Qb,Yl),
fronttoken(Qd,Qc,","),
fronttoken(Qe,Qd,Zl),
fronttoken(Question,Qe,")").
```

```
/* un exemple de metaconnaissance du prof */
/* representation non executable */


qap:-plus(X,Y,Z),Cote>8,write(X," plus ",Y," = ").
qap:-fois(X,Y,Z),Cote>8,write(X," fois ",Y," = ").
iad:-plus(X,Y,Z),Cote<8,write(X," plus ",Y," = Z").
iad:-fois(X,Y,Z),Cote<8,plusconnu,write(X," fois ",Y," = ",Z).
qaa:-plus(X,Y,?),X<2,Y<2.
qaa:-fois(X,Y,?),X<2,Y<2.
iaa:-plus(X,Y,Z),X<2,Y<2.
iaa:-fois(X,Y,Z),X<2,Y<2.
plusconnu:-Cote>8.
foisconnu:-Cote>8.
eval(X,Y):-X<Y,write("c'est ",Y).
eval(X,Y):-X=Y,write("c'est bon").
```

# BIBLIOGRAPHIE

1 : Module de formation 1 :"théorie de l'apprentissage",
FUNDP Namur, Unité de Pédagogie

2 : Conception assistée des applications informatiques :
1. Etude d'opportunité et analyse conceptuelle
F.Bodart et Y.Pigneur, Masson, Paris, 1983

3 : Turbo Prolog, the natural language of artificial intelligence
Borland Internationnal, First Edition, Scotts Valley,April 1986