Université de Nancy I

Centre de Recherche en Informatique de Nancy

80/718

Sc N° 80/5B

CONTRIBUTION A L'AMELIORATION DES PROCESSUS D'ENSEIGNEMENT, D'APPRENTISSAGE ET D'ORGANISATION DE L'EDUCATION. L'ORDINATEUR OUTIL ET OBJET DE FORMATION. APPLICATION AU PROJET SATIRE.

THESE DE DOCTORAT D'ETAT

ES SCIENCES MATHEMATIQUES

UNIVERSITÉ NANCY BIBLIOTHÈQUE

Soutenue le 22 Janvier 1980

par

Maryse QUÉRÉ

devant

Claude PAIR, Président du jury et rapporteur
Jean BERBAUM
Hélène BESTOUGEFF
Daniel COULON, rapporteur
Marion CRÉHANGE
Michael GRIFFITHS
Jean-Paul HATON
Jacques PERRIAULT, rapporteur
Michel SINTZOFF

Il est habituel de commencer la publication d'une thèse par un hommage reconnaissant aux différentes personnes qui en ont facilité la réalisation. Il n'est pas dans mon intention de me soustraire à cette tradition, et je voudrais même élargir plus que de coutume le champ de cet hommage.

Je voudrais tout d'abord remercier Reiji TAKAHASHI, qui a guidé mes premiers pas dans la recherche. Si j'ai plus tard abandonné les mathématiques dites pures, ce n'est certes pas à cause de lui.

Bertrand SCHWARTZ m'a fait entrer dans l'enseignement supérieur : le travail sous sa direction, au début d'une carrière, ne peut que l'orienter définitivement vers le goût de l'innovation pédagogique. Pierre ANTOINE et Olivier CLOUZOT, en me faisant connaître l'enseignement programmé, furent les acteurs initiaux du processus. François VIALLET, pour sa part, m'a aiguillée vers l'informatique : ce fut un tournant important. Avec eux, je remercie tous ceux qui, dans le complexe CUCES-ACUCES-INFA, m'ont assuré un environnement des plus agréables.

Claude PAIR mérite, bien sûr, une mention toute particulière : d'un côté il m'a poussée, avec son exigence et sa gentillesse habituelles, à démarrer ce travail et à le mener à son terme ; d'un autre côté, en me confiant d'importantes responsabilités d'animation tant à l'IUT qu'à l'IREM, il m'a incitée à distraire du temps que j'aurais pu consacrer à la recherche ; ce fut donc, d'une certaine façon, le meilleur et le pire des directeurs de recherche.....

UNIVERSITÉ NANCY BIBLIOTHÈQUE

.....mais ce meilleur et ce pire ayant tous deux été source de nombreuses satisfactions professionnelles, je lui exprime ici toute mon amicale gratitude.

Il m'est agréable d'y associer tous mes amis de l'IREM et de l'IUT avec qui le travail reste, après tant d'années, passionnant, et en particulier Marion CRÉHANGE et Michael GRIFFITHS, qui confirment cette amitié en étant membres du jury.

L'amour de la didactique étant une des choses au monde les plus mal partagées, j'ai hélas œuvré trop solitairement à ce travail. Cependant Jacques JARAY m'a donné des coups de main épisodiques mais efficaces dans le labyrinthe des systèmes d'exploitation. Jacques DUCLOY et Christine PISTER ont facilité mes rapports avec SOCRATE. Qu'ils en soient ici remerciés, ainsi que tout le personnel de l'IUCAL, qui m'a toujours aidée à régler dans la bonne humeur les problèmes inhérents à l'utilisation d'une machine. Merci également au groupe «POLLUX» pour sa participation, même involontaire, à la fabrication des cours d'algorithmique et de programmation.

Michel SINTZOFF, malgré la brièveté de son séjour à NANCY, est devenu indissociable de notre laboratoire. Bien que je n'aie pas directement travaillé avec lui, je lui suis reconnaissante d'avoir accepté de faire partie des membres du jury, dans lequel il apporte le point de vue du non-enseignant.

Hélène BESTOUGEFF, Daniel COULON et Jacques PERRIAULT se sont tous les trois, à un moment donné de leur vie, compromis avec l'enseignement assisté par ordinateur. Leur exemple et leurs critiques m'ont été très utiles. Merci à la première, dont la présence accroît de façon sensible la part féminine du jury. Merci au second pour avoir ainsi initialisé une collaboration qui, je l'espère, se poursuivra longtemps. Merci au troisième d'avoir distrait de son précieux temps pour une petite provinciale.

C'est encore à l'IREM que j'ai connu Jean BERBAUM. Je remercie cet expert en sciences de l'éducation d'affronter avec courage un tel aréopage d'informaticiens, ainsi que pour toutes les remarques qui ont contribué à améliorer mon premier manuscrit.

Je n'oublie pas l'UER de Sciences Mathématiques de NANCY I dont le soutien à cette thèse a pris plusieurs formes. Le secrétariat et la bibliothèque m'ont toujours été ouverts. C'est, bien sûr, Jean-Claude DERNIAME qui m'a appris à programmer. Jean-Paul HATON a accepté avec gentillesse ce rôle de frère aîné qui est un peu celui des membres du jury. Tous deux ont fait, à peu près en même temps que moi, leurs études à «l'institut», et il m'est agréable de travailler maintenant avec eux.

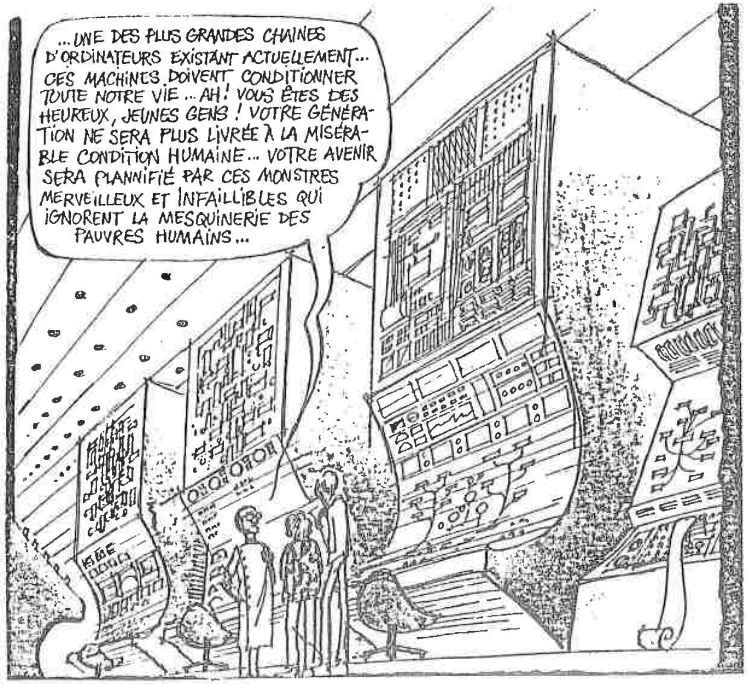
La chronologie de la fabrication d'une thèse fait que c'est toujours à la fin qu'on cite ceux et celles qui en ont assumé l'intendance. C'est pourtant eux qui méritent les remerciements les plus chaleureux !

Bravo donc à Nadine GAUTIER et Soizic DENDIEN (IUT), Christiane L'HOMMÉE et Marie-Thérèse WALTER (IREM) pour leur dactylographie exemplaire. Merci à Chantal GUILLAUMET (Cellule d'Information et d'Orientation de NANCY I) pour la photocomposition de la couverture et des têtes de chapitres. Merci enfin au service reprographie de l'ILF, et en particulier à monsieur RETOURNARD, pour l'impression de la couverture et pour la reliure.

Un homme sort de chez lui
C'est très tôt le matin
C'est un homme qui est triste
Cela se voit à sa figure
Soudain dans une boîte à ordures
Il voit un vieux Bottin Mondain
Quand on est triste on passe le temps
Et l'homme prend le Bottin
Le secoue un peu et le feuillette machinalement
Les choses sont comme elles sont
Cet homme si triste est triste parce qu'il s'appelle Ducon
Et il feuillette
Et continue à feuilleter
Et il s'arrête
A la page des D
Et il regarde à la colonne des D—U du.....
Et son regard d'homme triste devient plus gai plus clair
Personne
Vraiment personne ne porte le même nom
Je suis le seul Ducon
Dit-il entre ses dents
Et il jette le livre s'époussette les mains
Et poursuit fièrement son petit bonhomme de chemin.

Jacques Prévert.

UNIVERSITÉ NANCY BIBLIOTHÈQUE



SOMMAIRE

INTRODUCTION. 13

CHAPITRE 1. QUELQUES ASPECTS DE L'ACTIVITE EDUCATIVE. 19

1.1. Introduction. 21

1.2. L'apprentissage. 23

1.3. L'enseignement. 27

1.4. L'éducation processus cybernétique. Rôle de l'évaluation. 31

1.5. La gestion des activités éducatives. 33

CHAPITRE 2. UTILISATION DE L'ORDINATEUR A DES FINS PEDAGOGIQUES. 35

2.1. Introduction. 37

2.2. L'ordinateur instrument de laboratoire. 39

2.2.1. L'ordinateur en tant que calculateur. 39

2.2.2. L'ordinateur objet d'enseignement. 40

2.2.3. L'ordinateur outil de simulation. 41

2.3. L'ordinateur instrument de gestion pédagogique. 42

2.3.1. Les banques de données. 42

2.3.2. L'ordinateur dans le processus d'évaluation. 44

2.3.2.1. Analyse et traitement des réponses. 44

2.3.2.2. Génération et correction de questionnaires. 44

2.3.2.3. Evaluation individualisée. 44

2.3.3. La gestion de cheminement. 45

2.4. L'ordinateur instrument d'enseignement. 48

2.4.1. Les exercices répétés. 50

2.4.2. L'enseignement de type tutoriel. 51

2.4.3. Les enseignements de type non-directif. 54

2.4.3.1. La conception assistée par ordinateur. 55

2.4.3.2. L'enseignement assisté par ordinateur intelligent. 58

2.5. Conclusion 60

CHAPITRE 3. ANALYSE ET PROGRAMMATION D'UN CONTENU DIDACTIQUE. 61

3.1. Introduction. 63

3.2. Présentation et comparaison des applications. 66

3.3. Analyse d'un contenu. 67

3.3.1. Définition d'un contenu didactique. 67

3.3.2. Une analyse d'un contenu didactique. 69

3.3.2.1. Construction d'une ramification $\mathcal{G}$: présentation intuitive. 69

3.3.2.2. Rappels sur les ramifications. 71

3.3.2.3. Algorithme de transformation du graphe en ramification. 75

3.3.2.4. Fonctions pédagogiques attachées à l'analyse de contenu. 77

3.4. Programmation d'un contenu. 78

3.4.1. Définition. 78

3.4.2. Idée intuitive de l'algorithme. 80

3.4.3. Algorithme de programmation. 83

3.4.4. Vérification des contraintes. 84

CHAPITRE 4. ACQUISITION D'UN CONTENU DIDACTIQUE. 87

4.1. Introduction. 89

4.2. Modèle mathématique de l'enseignement programmé et de l'enseignement assisté par ordinateur. 90

4.2.1. Structure de l'item et du programme : une première formalisation. 90

4.2.2. Algorithmes d'enseignement. 93

4.2.3. Algorithmes d'enseignement de Markov. 95

4.2.4. Description des principaux types de programmes. 95

4.2.5. Automates d'enseignement pour A.E.M. 96

4.2.6. L'EAO de type tutoriel classique. 99

4.2.7. Les extensions possibles. Le macro-E.A.O. 99

4.3. Un système d'acquisition d'un contenu. 100

4.3.1. Objets. 101

4.3.2. Fonctions. 102

4.3.2.1. Prologue. 102

4.3.2.2. Mode documentaire. 103

4.3.2.3. Mode adaptation de cours. 103

4.3.2.4. Mode tutoriel. 103

4.3.2.5. Mode récapitulatif. 103

4.3.2.6. Epilogue. 103

4.4. Utilisation dans quatre types courants d'applications. 104

4.4.1. Interrogation de bases de données. 104

4.4.2. E.A.O. de type tutoriel classique. 104

4.4.3. Gestion de cheminement. 105

4.4.4. Exercices répétés. 105

CHAPITRE 5. LE PROJET SATIRE : UN LOGICIEL DE MACRO-ENSEIGNEMENT ASSISTE PAR ORDINATEUR UTILISANT UNE BASE DE DONNEES. 107

5.1. Historique du projet SATIRE. 109

5.2. Pourquoi une base de données ? 110

5.3. Les objets du projet SATIRE. 112

5.3.1. Les concepts. 112

5.3.2. Les leçons. 113

5.3.3. Les questions. 113

5.3.4. Les étudiants. 113

5.3.5. Les items. 114

5.3.6. Les comportements. 114

5.3.7. Les interventions. 114

5.4. Les fonctions du projet SATIRE. 115

5.4.1. Fonctions relatives à l'enseignant. 115

5.4.2. Fonctions relatives à l'étudiant. 115

5.4.3. Administration de la base de données. 121

CHAPITRE 6. DIDACTIQUE DE LA PROGRAMMATION ET UTILISATION DE SATIRE DANS SON ENSEIGNEMENT. 123

6.1. Introduction à la didactique de la programmation. 125

6.2. Les concepts et les différentes progressions possibles. 126

6.2.1. Les concepts. 126

6.2.2. L'apprentissage en spirale. 128

6.2.3. Langage ou pas langage ? 130

6.3. L'apprentissage par la résolution de problèmes. 130

6.4. Macro-enseignement assisté par ordinateur de la programmation. 133

6.4.1. Les concepts. 134

6.4.2. Les leçons. 134

6.4.3. Les questions et l'analyse des réponses. 137

Annexe 6A. Liste des concepts et relations entre les concepts. 139

Annexe 6B. Exemples de concepts. 154

Annexe 6C. Exemple de questions associées à un concept pour une leçon. 163

CHAPITRE 7. INTERFACE DE SATIRE AVEC LE SYSTEME SIRIS 8 et le SGBD SOCRATE II. Insuffisances et propositions. 165

7.1. Considérations didactiques et conséquences techniques. 167

7.2. Le système de gestion de bases de données SOCRATE II. 168

7.2.1. Structure logique de la base. 168

7.2.2. Mise en oeuvre de SOCRATE II. 170

7.2.3. Nature des items et inexistence d'une caractéristique adéquate. 171

7.3. Caractéristiques du programme de dialogue. 173

Annexe 7A. Description théorique de la structure de la base. 174

CHAPITRE 8. PROGRAMMATION DES DIFFERENTES FONCTIONS DE SATIRE. 177

8.1. Structure de la base de données finalement retenue. 179

8.2. Fonctions programmées dans le langage de requêtes SOCRATE II. 181

8.3. Construction assistée de leçons. 184

8.3.1. Premier algorithme d'analyse du contenu. 186

8.3.2. Premier algorithme de programmation du contenu. 186

8.3.3. Programme correspondant. 189

8.4. Introduction de leçons. 192

8.5. Dialogue étudiant-ordinateur. 192

8.5.1. Extrait du programme source. 192

8.5.2. Extrait des sous-programmes SOCRATE utilisés. 200

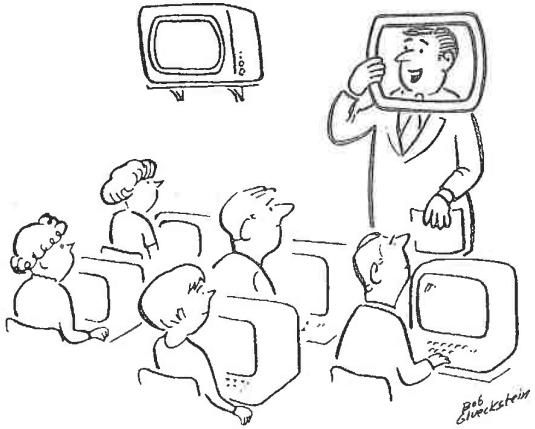
8.5.3. Extrait d'un dialogue. 204

8.6. Traitement des algorithmes écrits selon la méthode de programmation déductive. 206

CONCLUSIONS ET PERSPECTIVES. 207

BIBLIOGRAPHIE. 211

Les illustrations sont extraites de la revue Educational Technology.



"There, do I come across better now?"

INTRODUCTION

L'orientation que prend un travail de recherche relève toujours, pour une bonne part, du hasard : la bibliographie n'étant jamais exhaustive, la fréquentation de tel groupe de travail, la participation à tel colloque plutôt qu'à tel autre, précisent le centre d'intérêt. Dans un travail pluridisciplinaire, la façon dont s'est faite la formation (académique, "sur le tas", personnelle) met en relief tel ou tel aspect du travail. Enfin, si la recherche a pour objet l'enseignement, les activités d'enseignement du chercheur vont influencer considérablement son évolution. Bien sûr, quand la silhouette commence à prendre forme, le chercheur, adoptant une démarche plus scientifique, prend connaissance des travaux connexes et s'assure qu'il n'enfonce pas des portes ouvertes : mais en aucun cas le choix de son sujet ne se fait par l'examen préalable de "ce qui n'a pas encore été exploré".

C'est pourquoi il est important, pour présenter ce travail, de le situer dans sa genèse. Je prends ici le risque de faire apparaître clairement la longueur de cette dernière, mais peu importe, car ce travail ne serait pas ce qu'il est si le terme en avait été fixé plus tôt.

J'ai donc commencé par me former à l'enseignement programmé : j'ai traduit et adapté des cours programmés [31, 163], j'ai participé à la rédaction de certains autres [88, 89, 90, 91], j'ai assuré de la formation de formateurs [108], j'ai pris part à des recherches en sciences de l'éducation [38, 106, 107]. J'avais appris les mathématiques : j'ai été tout naturellement conduite à travailler sur des cours programmés de mathématiques. Tout aussi naturellement, j'ai eu envie de tenter une formalisation de ce qui m'apparaissait à l'époque relever d'un bricolage élémentaire, à savoir l'analyse et la programmation des contenus à enseigner. J'ai lu ce qui se faisait, essayant de faire la part des "bonnes" et des "mauvaises" choses. Le hasard a voulu que dans le laboratoire dont je faisais partie, les ramifications soient à la mode. Le chapitre 3 avait vu le jour. Un peu plus tard, il fut partiellement publié parce que, ne voyant pas le terme de mon travail, je souhaitais apporter rapidement une contribution méthodologique aux praticiens de l'enseignement programmé.

A l'époque, les ordinateurs, que je ne connaissais que pour y avoir "fait passer" des petits programmes FORTRAN ou ALGOL, m'apparaissaient comme des super-machines à tourner les pages des livres programmés, ce qu'ils étaient, mis à part les premières tentatives d'analyse de réponses en langue naturelle. On appelait cela l'enseignement assisté par ordinateur.

Au fait, pourquoi cet intérêt pour l'enseignement programmé ? Je trouvais déjà ce mode d'enseignement très rudimentaire, mais le fait que mes premiers étudiants furent des adultes engagés dans la vie professionnelle m'avait convertie à l'idée d'un enseignement par objectifs, individualisé, auto-évaluatif, dont le retour ultérieur à un enseignement universitaire plus classique n'a pu me détourner.

Alors j'ai eu envie de conserver les "bons côtés" de l'E.P. et d'explorer les voies nouvelles que l'ordinateur promettait d'apporter. Là encore ma formation de mathématicienne m'a aidée, ainsi qu'un nouveau hasard, à savoir la possibilité que j'ai eue d'étudier les travaux de Helmar FRANK et son équipe [63]. Une nouvelle phase de modélisation allait déboucher sur le chapitre 4, dont les premières bases furent présentées à Marseille en 1975 [169]. J'avais entre temps étudié un peu d'informatique, du moins suffisamment pour être certaine que le système proposé était réalisable, et pour affirmer que si ce système se faisait, il utiliserait pour stocker la connaissance une base de données. Encore ce dernier concept restait-il pour moi assez flou.

Ce principe de "faisabilité" étant posé, fallait-il faire ? Je n'étais pas tellement motivée : il y avait déjà une telle profusion de systèmes, l'ordinateur des universités de Nancy ne s'y prêtait pas trop, il faudrait programmer... Voilà que mon intérêt s'envole ailleurs, vers la didactique des mathématiques : d'accord pour transmettre de façon tutorielle des connaissances par ordinateur, mais l'ordinateur fait des choses tellement plus "chouettes"... il démontre des théorèmes, résout des équations trigonométriques, calcule des primitives : voilà des nouvelles voies à explorer [77, 168]. D'ailleurs, dans les autres disciplines, on s'agite de la même façon, et l'enseignement assisté par ordinateur est mis, provisoirement, aux oubliettes. Tout juste mérite-t-il des conférences ou enseignements de 3e cycle pour les maîtres ou élèves-maîtres de l'enseignement secondaire !

Mais pour expérimenter des secteurs "de pointe", encore faudrait-il que le logiciel de base existe ! Alors... on demande des crédits. Pierre accueille favorablement la demande, mais, se déclarant incompétent, la renvoie à Paul. Pendant ce temps, je me lasse, je vais voir du côté des recherches sur la "compréhension", qui incluent des expériences d'enseignement assisté par ordinateur (ce qui me confirme qu'il s'agit encore pour le moment d'une direction dont le point de rencontre avec la mienne n'est pas proche). Je me mets à enseigner la programmation et à flirter avec MEDEE [146]. Paul enfin accepte, en l'infléchissant, une partie du projet : il prend donc sa forme définitive, avec la chance inespérée de l'arrivée de SOCRATE II, qui permet l'achèvement des chapitres 5 et 7 dans des conditions pas trop mauvaises. C'est alors que je prends conscience que l'enfant n'est plus de l'enseignement assisté par ordinateur, mais un intermédiaire entre l'E.A.O. et l'enseignement géré par ordinateur, intermédiaire qui peut prendre l'une ou l'autre forme selon l'interprétation qu'on en fait. Une des interprétations possibles est disponible dans le prototype de cours qui est développé au chapitre 6.

Mathématicienne, informaticienne, enseignante... je ne suis ni psychologue ni spécialiste en sciences de l'éducation. Que ceux-ci, que j'ai par goût beaucoup fréquentés, me pardonnent les erreurs et omissions des parties qui les concernent.

On a dit plus haut que ce travail était pluridisciplinaire. Je vais en faire une brève présentation en indiquant la dominante de chacun des chapitres. Entendons-nous bien : je ne vais pas ici parler du contenu, n'en déplaise à ceux qui, dans une thèse, ne lisent que l'introduction et la conclusion. En effet, j'estime que pour comprendre ce travail, il faut être spécialiste du domaine, ou avoir lu auparavant les chapitres 1 et 2 : c'est donc dans la conclusion du chapitre 2 que le contenu est présenté, ainsi que dans les introductions des chapitres 3, 4 et 5. Le lecteur impatient peut, s'il le désire, s'y reporter...

Le premier chapitre est plutôt orienté vers les sciences de l'éducation, ainsi que le début du chapitre 6. Les idées qui y sont développées sont largement inspirées par la pratique du métier, s'appuient sur des résultats expérimentaux (recherches sur l'enseignement menées dans des organismes comme l'I.N.R.P. ou les I.R.E.M.) et ont parfois reçu l'éclairage de théories élaborées par des psychologues, du moins celles qui ont un rapport avec l'enseignement programmé ou l'apprentissage des mathématiques.

Le chapitre 2 se veut une synthèse de l'état de l'art.

Les chapitres 3 et 4 sont un essai de formalisation, à partir d'outils mathématiques tels que les ramifications, les algorithmes et les automates, d'une partie des activités décrites dans les chapitres 1 et 2. Le chapitre 5 précise le travail dans un but de développement de logiciel.

Le chapitre 6 relève de la didactique de la programmation en même temps qu'il permet une validation des idées précédemment exprimées.

Les chapitres 7 et 8 sont un travail d'analyste-programmeur, avec des objectifs d'économie de moyens (utilisation maximale de logiciels existants), de fiabilité et de portabilité.

Ce travail étant proposé comme un travail de mathématiques, sans doute est-il pertinent de préciser d'entrée de jeu un peu de vocabulaire [110]. Actuellement, le mot didactique est surtout utilisé comme synonyme de pédagogie. Pourtant son emploi privilégie les aspects particuliers suivants de la pédagogie :

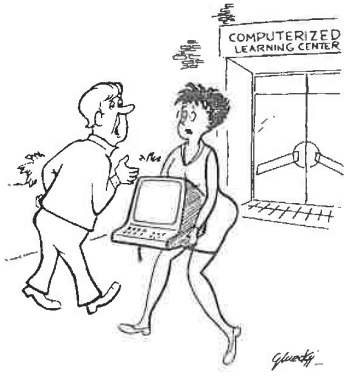
- l'aspect cognitif
- l'aspect technologique
- l'aspect relatif (on parle de didactique de telle ou telle discipline).

En fait, la didactique ne constitue ni une discipline, ni une sous-discipline, mais une démarche, c'est-à-dire un certain mode d'analyse des phénomènes de l'enseignement.

Une étude didactique suppose fréquemment l'intervention d'au moins deux disciplines : d'une part celle dont on considère l'enseignement (discipline-objet), et d'autre part une discipline intervenant comme outil dans l'étude envisagée. Il se

trouve ici qu'un même groupe de disciplines joue à la fois le rôle d'objet et d'outil : les oppositions ne s'en trouvent nullement réduites pour autant ! Les informaticiens qui prêchent la prééminence de l'informatique dans toutes les activités humaines ne verront peut-être pas d'un oeil serein des propositions visant à l'introduire dans l'enseignement de l'informatique.

« J'ai comme l'impression que les choses vont changer cette année ».



"Homework?"

CHAPITRE 1 : Quelques aspects de l'activité éducative

1.1. INTRODUCTION.

Il n'est pas question, dans ce chapitre, de se lancer dans une description complète des processus mis en oeuvre dans un système éducatif, ni de faire le point sur les différentes conceptions dans le domaine. Nous voulons seulement exposer, outre un minimum de terminologie, les courants qui nous ont le plus influencée dans notre tentative de modélisation et d'introduction de l'ordinateur comme agent du système. Toute modélisation et toute mécanisation appauvrissant nécessairement la réalité des processus éducatifs, nous tenons à faire ici un bref catalogue, même incomplet, et, dans la suite de l'ouvrage, nous y ferons référence, afin de n'être pas accusée de placer l'informatique *avant* toute chose, alors que nous plaçons l'informatique *parmi* les autres choses.

Nous utiliserons, pour cette introduction, une terminologie empruntée à LANSKY [114]. Précisons tout de suite que les diverses classifications qui apparaîtront dans cette thèse sont bien sûr des exercices de style, en général apparues a posteriori, mais qu'elles ont le mérite de ne rien laisser de côté, c'est pourquoi nous les utilisons.

Dans ce qui suit, le mot éducation doit être compris dans son sens le plus large, incluant tous les niveaux de formation, générale ou professionnelle, initiale ou continue. A ce titre, les citations qui suivent emploient indifféremment le mot enfant, le mot adulte, le mot étudiant. Ce n'est qu'à partir du chapitre 5 que le public visé est un public se situant au niveau post-secondaire.

Nous distinguerons donc dans l'éducation trois points de vue, respectivement appelés microscopique, intermédiaire et macroscopique, et dans chacun de ces points de vue l'objet, le sujet et l'instrument. La structure d'un système éducatif, que nous détaillons ci-après selon ces critères, se trouve résumée dans le schéma de la figure 1.

. point de vue microscopique.

L'unité de base est le système d'étude (SyEt), comprenant :

- l'objet de l'étude (OEt), par exemple la matière.
- l'instrument de l'objet d'étude (IOEt), le livre ou un quelconque medium.
- le sujet d'étude (SEt), par exemple l'écolier.
- l'instrument du sujet d'étude (ISEt), par exemple le crayon, le papier, la calculette.

L'interaction spécifique entre OEt et SEt est ce qu'on appelle classiquement une *situation d'apprentissage* (l'apprentissage étant le changement de

comportement ou d'attitude qui accompagne cette interaction).

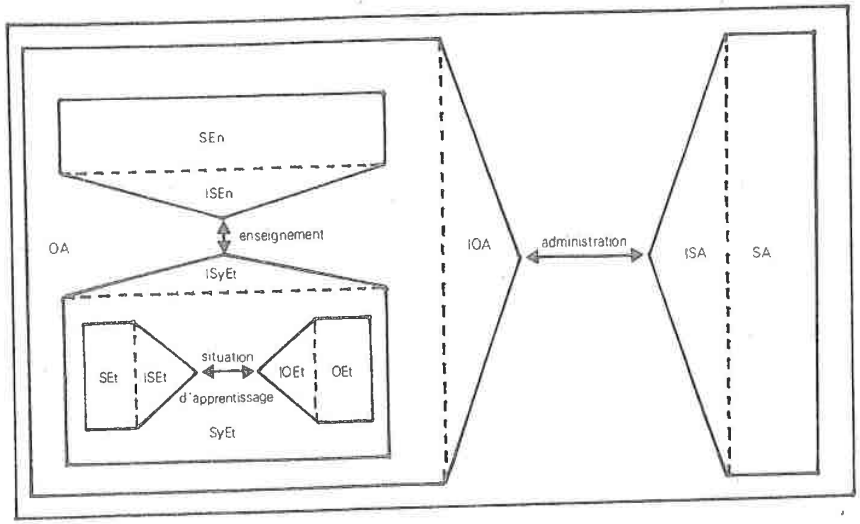


figure 1 : Structure d'un système éducatif selon LANSKY [114]

. point de vue intermédiaire.

L'unité de base est ici le système d'enseignement (OA), comprenant :

- l'objet d'enseignement, égal au système d'étude.
- l'instrument de l'objet d'enseignement (IOEn), par exemple bibliothèque, médiathèque.
- le sujet d'enseignement (SEn), par exemple le professeur.
- l'instrument du sujet d'enseignement (ISEn).

L'interaction est ici *l'enseignement* conçu comme un contrôle du SyEt par le SEn.

. point de vue macroscopique.

L'unité de base est le système d'administration de l'enseignement, comprenant :

- l'objet de l'administration, égal au système d'enseignement.
- l'instrument de l'objet de l'administration (IOA), par exemple l'horaire des écoles.
- le sujet d'administration (SA), président d'université ou inspecteur général.
- l'instrument du sujet d'administration (ISA), données, formulaires,...

L'interaction est alors *l'administration* conçue comme un contrôle du SEn par le SA.

On comprend que ce soit les deux premiers niveaux qui ont le plus passionné les pédagogues, psychologues et autres chercheurs en éducation. Cependant le troisième mérite également d'être étudié : l'introduction de l'ordinateur, avec ses grandes capacités de mémorisation, permet en partie de combler le retard, en même temps qu'il constitue un instrument de recherche pour les deux premiers.

Il est un peu artificiel de séparer ces différents points de vue. Cependant, pour la clarté de l'exposé, nous distinguerons les idées relatives à l'apprentissage de celles qui sont plutôt spécifiques de l'enseignement, nous réservant d'en faire ensuite une synthèse. L'aspect cybernétique de l'éducation sera ensuite abordé, avec une mention particulière de l'activité d'évaluation. Nous terminerons enfin ce chapitre par un paragraphe consacré à la fonction d'administration.

1.2. L'APPRENTISSAGE.

Les théories de l'apprentissage n'en sont encore qu'à leur balbutiement. Deux grands courants se sont jusqu'à présent opposés [112], le behaviorisme et la théorie opératoire. Une mention particulière peut être faite des psychologues soviétiques [193] qui se sont plus spécialement intéressés à l'acquisition des algorithmes.

La *théorie empiriste* ou *behavioriste* de SKINNER admet que la connaissance vient d'abord d'une information sensorielle, transmise de l'extérieur par le canal des sens. La connaissance étant extérieure à l'individu, celui-ci se l'approprie progressivement au fil de ses expériences. On en déduit parfois que, par des méthodes appropriées, tout individu peut acquérir telle ou telle notion.

La pédagogie qui en découle se base sur la notion de stimulus-réponse-renforcement et cherche à obtenir que, à stimulus égal, la réponse soit identique.

Ces pédagogies sont de type associationniste, c'est-à-dire que la connaissance du niveau supérieur est considérée comme la somme des acquisitions antérieures. Cette manière de voir parait souvent séduisante au pédagogue : il suffirait de bien programmer les stimuli pour que les connaissances s'accroissent régulièrement.

La *théorie opératoire* de PIAGET, elle, s'appuie sur la notion d'interaction entre le sujet et l'objet. Le sujet donne une signification à l'objet qui "répond" de manière plus ou moins satisfaisante par rapport aux schèmes propres du sujet.

L'intelligence se construit dans la mesure où l'expérience nouvelle ne vient pas simplement s'ajouter à l'acquis antérieur mais provoque une réorganisation, une restructuration de l'acquis en une totalité cohérente.

Diverses améliorations ont par la suite été proposées à la théorie behavioriste, selon une approche pragmatique [3] : l'atome stimulus-réponse devenant un "morceau de comportement" pouvant être grand ou très petit. Cette approche donnera lieu au courant qu'il est convenu d'appeler pédagogie par objectifs, et dont nous parlerons dans le prochain paragraphe.

Même en se limitant au strict plan des acquisitions, la critique de l'associationnisme est facile [185] : c'est une caractéristique des types supérieurs d'apprentissage que la dépendance des acquisitions les plus récentes des acquisitions précédentes. Une théorie de l'apprentissage doit traiter non seulement de processus de la pensée pour eux-mêmes, mais de processus organisés et interconnectés dans un système élaboré.

C'est donc quand on aborde le problème de la *compréhension* [1] que les partisans de la théorie opératoire semblent l'emporter. Le simple stimulus-réponse ne peut en effet rendre compte de processus aussi élaborés que ceux qui consistent à dégager invariants opératoires, règles d'action, calcul relationnel, tels qu'ils sont définis dans [71], ou semblent parfois récupérés par les courants issus du behaviorisme [64] :

- un *invariant opératoire* est un concept, un préconcept ou éventuellement un pseudo-concept. L'expression d'invariant opératoire permet de mettre en évidence, d'une part le fait que se constitue en objet logique stable (invariant) pour le sujet une classe de phénomènes soumis auparavant à variation, d'autre part que le critère de l'acquisition d'un invariant est l'action (opératoire) ou les réponses d'un sujet en situation. Le volume, le poids sont des invariants opératoires. Notons qu'un concept opératoire ne se laisse pas enfermer dans une définition, mais qu'il comporte en fait toutes ses formes de fonctionnement scientifiques et idéologiques. Les représentations [204] sont des formes déviées d'invariants opératoires s'appuyant sur des raisonnements spontanés parfois d'origine idéologique.

- une *règle d'action* est une règle qui permet d'engendrer des actions en fonction des valeurs prises par certaines variables de situation. Ces variables peuvent évoluer dans le temps, ou d'une situation à une autre ; une règle (ou une conjonction de règles) n'en est pas moins utilisée pour toute une classe de situations. Un algorithme de dénombrement d'objets peut être considéré comme une règle d'action.

- un *calcul relationnel* est une déduction (ou plus généralement une inférence) faite par le sujet en situation, en fonction des invariants opératoires dont il dispose dans le champ de situations considérées. Il peut être décrit, en général, par une composition ou une transformation de relations et de propriétés. La proportionnalité entre l'allongement d'un ressort et la mesure de la masse qui y est accrochée est un exemple d'une telle inférence.

Comme nous l'avons vu pour les invariants (cas des représentations), règles d'action et calculs relationnels peuvent être adéquats ou inadéquats. Ils n'en jouent pas moins le même rôle fondamental, qui est de refléter la réalité matérielle au plan symbolique et de régler l'affectivité du sujet, cette activité étant elle-même la source de la représentation cognitive.

L'activité cognitive consiste aussi, dans une large mesure, dans l'exploration du champ de connaissances, dans le rapprochement analogique ou contradictoire de connaissances différentes, et dans l'effort pour rendre plus universel, plus cohérent et plus économique le système cognitif ainsi exploré.

Pour parler un peu des instruments de l'apprentissage, mentionnons ici le statut particulier de l'*activité expérimentale* [41], qui consiste à élaborer une formulation (ou modèle) prise comme hypothèse, à la soumettre à vérification en la confrontant à une réalité qui n'est pas mise en question et à se prononcer sur la validité de cette hypothèse à la lumière des résultats de cette confrontation.

Nous parlerons plus longuement de cette activité expérimentale dans le chapitre 2, à propos de la simulation, et de l'activité de l'étudiant en général dans le chapitre 6, à propos de l'apprentissage par résolution de problèmes.

Un autre aspect important des phénomènes de compréhension est celui de la *discontinuité* dans l'apprentissage : le développement de l'intelligence se présente comme une espèce de fonction à seuil. C'est ainsi que le développement génétique comporte des étapes successives, des stades (réflexes, perception et habitudes, sensori-moteur, pré-opératoire ou figural, opératoire concret, opératoire formel). C'est ainsi qu'une hypothèse actuelle [202] affirme que l'adulte dispose de registres de fonctionnements, correspondant approximativement aux stades de PIAGET, et que confronté dans une matière à une situation donnée, il peut mettre en oeuvre un registre mal approprié, aboutissant ainsi à une erreur : l'apprentissage consiste alors à rectifier le raisonnement à propos de la matière, c'est-à-dire à changer de registre de fonctionnement. La figure 2 indique la formation chez l'enfant, puis la coexistence chez l'adulte, de tels registres de fonctionnement. On remarquera que le schéma reste ouvert sur la droite, pour ne pas préjuger d'un arrêt de la genèse des instruments cognitifs.

Terminons ce paragraphe par le problème extrêmement important du *langage* [71,129].

Après la sortie de l'enfance, l'apprentissage chez l'étudiant se fait essentiellement à travers la lecture et l'écoute, soit le langage écrit et parlé... Ce rôle fondamental des signes implique que le but de l'apprentissage est de comprendre un "signifié" à travers un message signifiant. Or l'analyse faite ci-dessus de la

représentation cognitive se double de la distinction fondamentale entre signifiant et signifié qui est de la plus grande importance pour l'acquisition des connaissances scientifiques. En effet, invariants, règles d'action et calcul relationnels se situent au plan du signifié, mais ce signifié fait l'objet, dans la communication et dans la pensée, d'une explicitation symbolique partielle : schémas, dessins, algèbre, langage naturel. Plusieurs systèmes de signifiants coexistent ainsi, avec leur syntaxe et leur sémantique propres. Ils ne sont pas le signifié. Ils ne renvoient pas directement à la réalité matérielle mais au signifié. Ils renvoient aussi les uns aux autres. Ce problème débouche très directement sur les méthodes d'évaluation. Citons pour exemple l'histoire racontée par Stella BARUK [8] concernant un élève qui connaissait par coeur et semblait avoir compris la définition de l'application bijective, mais qui en donnait une illustration par diagramme sagittal absolument contraire, tout simplement parce que sa réalité familiale et sociale conditionnait faussement sa compréhension du mot "unique".

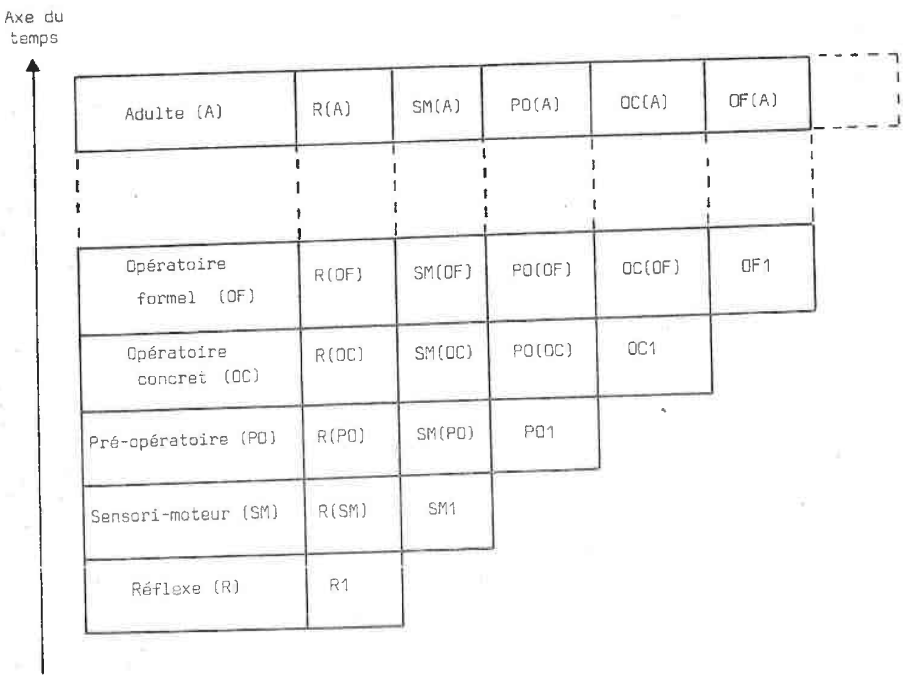


figure 2 : Les registres de fonctionnement disponibles [202]

1.3. L'ENSEIGNEMENT.

La plupart des programmes d'enseignement, encore appelés curricula, comportent en général deux parties : la première, souvent exprimée en termes de finalités, concerne le développement de l'enfant ou plus généralement de l'individu. La seconde est plus prosaïquement une liste de connaissances le plus souvent partitionnée en plusieurs matières [1].

Pour certains, comme l'a déploré le professeur LICHNEROWICZ, le but de l'enseignement est "la transmission des connaissances acquises, comme s'il s'agissait d'un capital positif bien rangé dans les cases d'un coffre-fort".

C'est un des mérites de la théorie behavioriste que d'avoir mis l'accent sur le fait que l'acquisition des "connaissances" devait se mesurer en termes de modification de comportement observable. De plus en plus, on voit se développer des tentatives pour formuler un curriculum en termes d'*objectifs opérationnels* [48].

"Le mot objectif, qui dans le sens de "but à atteindre" est apparu à la fin du XIXe siècle, a été et reste attaché à des degrés de généralité très divers qui ont fait perdre à ce mot toute sa précision. Il faut avant tout distinguer les finalités (buts à long terme), les options (choix de méthodes, de stratégies) des objectifs qu'on pourrait appeler opératoires ou opérationnels (on peut dire encore "formulés en termes de comportement")".

De nombreux chercheurs ont proposé des taxonomies d'objectifs [17, 64] . Cela va de la simple constatation pragmatique [135] :

"Il y a corrélativement plusieurs niveaux de procédés de l'esprit, c'est-à-dire des démarches régulièrement observables :

- idée de recette (maîtrise parfaitement éprouvée d'un processus qui a réussi)
- idée d'algorithme (séquence d'opérations qui occupe l'esprit pendant tout le temps de sa réalisation)
- idée de méthode
- attitudes d'esprit"

à des listes élaborées et structurées. Citons pour mémoire :

- les opérations "d'entraînement mental" de Peuple et Culture
  "énumérer, décrire, comparer, distinguer, classer, définir, dégager les aspects, discerner les points de vue, dégager les contradictions, situer dans le temps, situer dans l'espace, déterminer les causes, préciser les conséquences, dégager des lois, se référer aux théories, préciser les principes, préciser les buts, connaître ou prévoir les moyens, prévoir les méthodes".
- le modèle tri-dimensionnel ou cube de GUILFORD (figure 3), qui dans trois directions appelées contenus, opérations et produits, suggère une hiérarchie des

schémas de la pensée [112]. Un point de l'espace repéré par ses trois coordonnées correspond donc à un objectif, par exemple la "mémorisation de faits symboliques".

- la classification NLSMA [197] s'appliquant aux énoncés d'exercices mathématiques, et qui distingue les faits spécifiques des concepts, ainsi que différents niveaux de raisonnement.

A
1

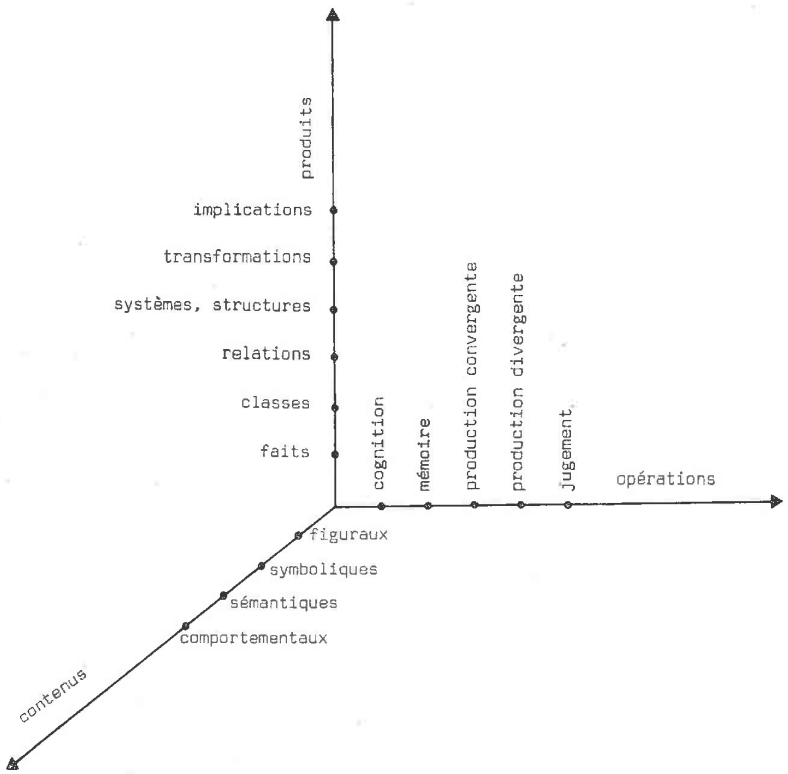


figure 3 : Le cube de GUILFORD [112]

Un *fait spécifique* est en quelque sorte un atome de connaissance. Parmi les connaissances, les faits spécifiques sont caractérisés par la propriété d'être isolément mémorisables et formulables, c'est-à-dire de pouvoir être exprimés en une phrase (française ou mathématique) simple (i.e. sans subordonnée).

Un *concept* est un ensemble de faits spécifiques.

La classification contient quatre niveaux A, B, C et D concernant le domaine cognitif, et deux niveaux E et F concernant le domaine affectif.

Niveau A : La connaissance et l'utilisation des faits spécifiques mémorisés.
- A.1. Connaissance des faits spécifiques.
- A.2. Connaissance de la terminologie.
- A.3. Aptitude à effectuer des algorithmes.

Niveau B : La connaissance et l'utilisation des concepts mémorisés.
- B.1. Connaissance des concepts.
- B.2. Connaissance de principes, règles, généralisations.
- B.3. Connaissance des structures mathématiques.
- B.4. Aptitude à traduire un énoncé d'une formulation à une autre.
- B.5. Aptitude à suivre un raisonnement.

Niveau C : Les applications.
- C.1. Aptitude à résoudre des problèmes routiniers.
- C.2. Aptitude à comparer, ordonner.
- C.3. Aptitude à interpréter des données.
- C.4. Aptitude à reconnaître des relations (exemples : périodicité, symétrie,... reconnaissance de formes)

Niveau D : La découverte.
- D.1. Aptitude à résoudre des problèmes inhabituels.
- D.2. Aptitude à découvrir des relations.
- D.3. Aptitude à démontrer.
- D.4. Aptitude à critiquer la validité d'un raisonnement.
- D.5. Aptitude à formuler et valider des généralisations.

Niveau E : Attitudes et intérêts (motivation, appréhensions, goûts personnels).

Niveau F : Appréciation (utilité des mathématiques, valeur de l'enseignement mathématique, élaboration).

Le problème de la détermination des objectifs étant supposé résolu, encore faut-il déterminer la nature de l'enseignement. Quittant le domaine de la didactique, on aborde là celui de la *pédagogie*. C'est la deuxième fois que nous faisons cette distinction (voir la fin de l'introduction) : alors que la didactique s'occupe

d'une discipline, ou d'un groupe de disciplines, - et on notera que les exemples qui précèdent se rapportent tous aux disciplines scientifiques, qui sont les seules que nous connaissons un peu -, la pédagogie traite de l'organisation de l'enseignement d'une manière souvent indépendante de la discipline.

Comme les théories de l'apprentissage, les pratiques pédagogiques s'opposent souvent [79].

Pour certains, l'acte pédagogique a pour objet essentiel l'apprentissage d'un "modèle" défini par l'adulte à partir de la logique et de la structure de la discipline : cette dernière détermine la nature et l'ordre des apprentissages. La pédagogie sera qualifiée d'"active" lorsque le maître prend en compte les intérêts et le développement des enfants en s'appuyant sur le dialogue orienté constamment par l'objectif à atteindre grâce à une maïeutique socratique. Elle accèdera au statut de "méthode de découverte" lorsque les enfants auront au début de l'exercice la possibilité de travailler librement sur un matériel spécialement choisi par le maître - souvent spécialement construit - en vue de l'objectif à atteindre.

D'autres au contraire cherchent à stimuler le dynamisme de l'enfant qui le pousse à apprendre lorsqu'il est placé dans un environnement social favorable et à l'orienter vers la découverte et la réalisation des objectifs pédagogiques reconnus comme souhaitables par l'institution scolaire...

Les défenseurs de la première approche insistent sur le fait que les sciences ne peuvent pas être découvertes par un coup de baguette magique alors qu'elles sont des constructions très progressives des sociétés humaines et que les obstacles à la connaissance n'ont pu être surmontés que par la découverte de situations de départ privilégiées. De plus, chaque discipline est caractérisée par une structure qui définit les rapports nécessaires entre concepts... Inversement, les partisans d'une pédagogie de la construction du savoir par une investigation autonome se basent sur les données de la psychologie génétique (en particulier de l'école de PIAGET) pour affirmer que l'enfant ne peut surmonter les représentations subjectives par une simple présentation des "modèles" adultes même s'il s'accompagne de la "purge" de certaines représentations spontanées grâce à des contre-exemples bien choisis...

En fait, les pratiques pédagogiques concrètes ne s'opposent pas de façon aussi tranchée. Il est probable que, lorsque l'objectif général vise la mise à disposition de certains algorithmes ou cherche à accroître la vitesse d'exécution à propos de savoir-faire - aussi bien lorsqu'il s'agit de connaissances relativement complexes comme l'assimilation d'une méthode de résolution d'un certain type de problème que dans les cas plus simples dont l'exemple type est constitué par les opérations arithmétiques - des séquences d'apprentissage bien construites peuvent

avoir une certaine efficacité [112]. De même une pédagogie basée sur la théorie opératoire pourra s'appuyer sur une définition précise des objectifs.

Reste à résoudre le problème de l'ordre dans lequel le professeur va planifier la poursuite des objectifs. Là encore les deux points de vue, après s'être vivement opposés, se rejoignent [1] : chacun sait qu'on ne peut pas faire un cours de mathématiques véritables en suivant les méthodes de rédaction de BOURBAKI et que, inversement, l'analyse de cas particuliers en aussi grand nombre qu'on veut ne suffit pas à définir le cas général (sauf, bien sûr, dans un domaine fini). Ici encore, c'est une dialectique qui fait progresser la compréhension.

1.4. L'EDUCATION PROCESSUS CYBERNETIQUE. ROLE DE L'EVALUATION.

Le modèle de LANSKY décrit plus haut a le défaut de se limiter à une description statique du système éducatif. Un autre point de vue, complémentaire, consiste à considérer que, comme toute activité visant un but ("conduire un élève à la connaissance visée"), l'activité scolaire peut être décrite dans les termes du modèle cybernétique et de ses boucles d'adaptation successives, selon le schéma de la figure 4, emprunté à CARDINET [30]. Une telle approche, souvent appelée systémique, est la plupart du temps utilisée pour simuler le processus éducatif. Noter que le point de départ, sur le schéma, est le point A , et qu'on sort du circuit par le point B après un certain nombre de passages dans les boucles d'adaptation.

Ce qui nous intéresse dans ce schéma, ce sont justement ces deux boucles d'adaptation, qui reposent sur un processus d'évaluation, au sens du second rôle que lui assigne MIGNE [133] :

1. Elle est une activité de recherche par laquelle le formateur s'efforce de connaître la situation sur laquelle il agit et les résultats de cette action, les modifications entraînées par son intervention en les distinguant des modifications liées à des facteurs extérieurs à elle.

2. En même temps, l'évaluation n'est qu'un moment de l'action pédagogique sur laquelle elle porte. Elle est un processus d'action-recherche. Elle n'est donc pas une pure activité de recherche à prétention objectiviste mais en tant que processus de recherche elle modifie l'action sur laquelle elle porte.

Considérant qu'on peut évaluer par rapport

(a1) aux objectifs de la formation
(a2) à des instruments d'évaluation
(a3) à l'individu
(a4) au groupe de référence,

et que l'évaluation peut porter sur

(b1) la pratique pédagogique du formateur

(b2) le programme de formation

(b3) l'individu

(b4) le groupe

(b5) l'organisme de formation,

la boucle d'adaptation la plus interne se rapporte à (a3), (b1), (b3) ou (b4) et la boucle la plus externe à (a1), (b2) et (b5).

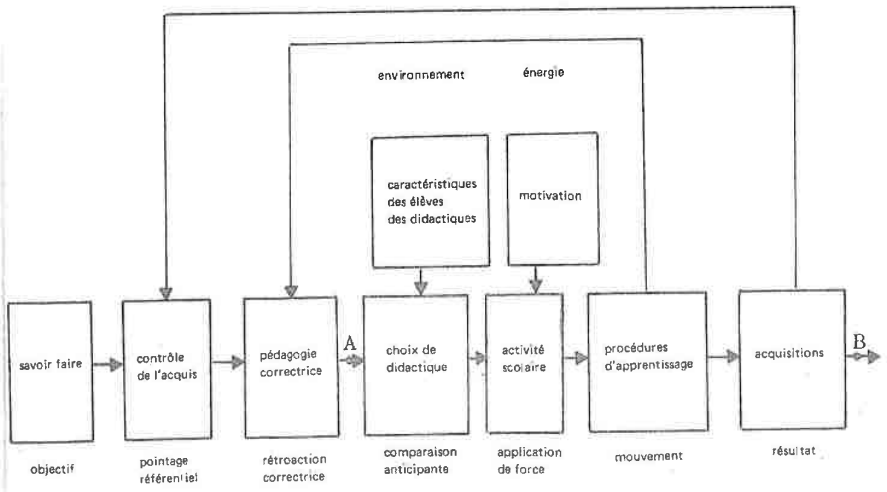


figure 4 : L'éducation processus cybernétique [30]

1.5. LA GESTION DES ACTIVITES EDUCATIVES.

On trouve dans le schéma de LANSKY les objets qui doivent être mémorisés par le système : il s'agit des instruments des objets d'étude regroupés dans l'instrument de l'objet d'enseignement, éventuellement des instruments des sujets d'étude (ressources) et des sujets d'enseignement, enfin des instruments de l'objet et du sujet de l'administration.

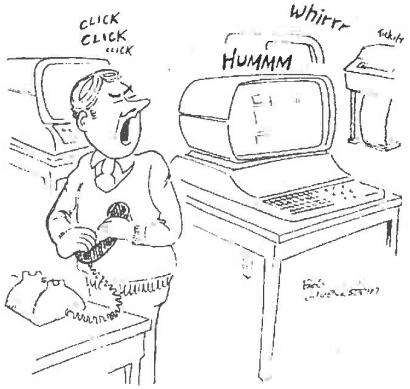
Tout ceci constitue un *système de données* [47], dans le sens d'un ensemble de moyens pour acquérir, interroger et modifier des données.

L'acquisition d'une donnée est le fait de la faire passer d'un sujet externe (feuille d'examen remplie par un étudiant) à un support interne au système (archives du service des examens).

Interroger une donnée, c'est chercher à en extraire des renseignements (quelle est la liste des étudiants inscrits au module M1 en 1979/80 ?).

Modifier une donnée est facile à comprendre : prenons l'exemple d'un changement de manuel de géographie dans un établissement secondaire.

Dans une pédagogie traditionnelle de style magistral, comme on en rencontre encore beaucoup dans nos universités, cette activité de gestion peut sembler dérisoire. Nous verrons au chapitre suivant l'importance qu'elle peut prendre dans des systèmes éducatifs s'appuyant, par exemple, sur la notion d'objectif.



"QUIET!!!"

CHAPITRE 2 : Utilisation de l'ordinateur à des fins pédagogiques

2.1. INTRODUCTION.

L'utilisation de l'ordinateur dans l'éducation, à cause de la nature de ce dernier, pose en amont des problèmes de formalisation des processus qui ont été décrits dans le chapitre précédent. Il est d'ailleurs également possible de formaliser a priori [198], dans le but d'améliorer la mesurabilité et la reproductibilité des systèmes, l'incorporation de l'ordinateur et d'outils procéduraux devenant une conséquence possible mais non obligatoire. Ce qui d'emblée se prête à une formalisation : la structure de la matière, l'évaluation, la gestion des matériaux d'enseignement et des activités, le processus d'apprentissage.

Il est possible d'aller plus loin en réutilisant la description initiale du chapitre 1 pour cerner l'étendue de ce qu'il sera convenu d'appeler l'*informatique éducationnelle* [114] :

- OEt : l'ordinateur comme objet d'enseignement (computer science)
- IOEt : l'ordinateur comme medium, qui recouvre presque tous les projets d'enseignement assisté par ordinateur au sens strict
- SEt : simulation de l'étude par ordinateur (didactique formelle)
- ISEt : ordinateur instrument de laboratoire, par exemple dans sa fonction de calcul, ou de gestionnaire de banques de données...
- SyEt : simulation par ordinateur du système d'étude (didactique formelle)
- IOEn : système documentaire, banques de données, systèmes multi-média
- SEn : gestion de l'enseignement par ordinateur
- ISEn : l'ordinateur comme instrument de contrôle, de tests, d'aide à la préparation de l'enseignement
- OA : simulation du système d'enseignement par ordinateur
- IOA : organisation d'école assistée par ordinateur
- SA : gestion par ordinateur : paiement des salaires, planification d'examens...
- ISA : aide à la gestion (par exemple mémorisation de données)

Notons tout de suite que de nombreux exemples actuels d'informatique éducationnelle peuvent être mieux caractérisés, quant au fond, par différentes combinaisons de ces différents points. C'est ainsi que la plupart des projets d'enseignement assisté par ordinateur au sens strict sont caractérisés par la combinaison de IOEt, ISEt, ISEn et SEn.

De nombreux autres auteurs [44, 45, 62, 74, 103, 117, 131, 155, 156, 157, 173, 174] ont adopté pour faire leurs classifications une démarche plus pragmatique, ainsi qu'il convient pour un sujet où la "théorie" a souvent suivi la pratique. Comme le faisait remarquer TENZCAR à qui on posait la question de savoir si l'ordinateur

avait un avenir dans la formation [174], "mais... c'est une très bonne question et je suis convaincu que lorsqu'on a inventé le livre, il y a des gens qui se sont posé la question : quel est l'avenir du livre dans l'enseignement ? Et ils ont certainement organisé des colloques pour examiner ensemble l'avenir du livre dans l'enseignement... Et il y a même des doctorats qui ont été faits là dessus".

Cependant les résultats auxquels ces auteurs arrivent recoupent les précédents, avec sans doute un peu moins de finesse. Comme eux, nous nous sommes limités à l'utilisation de l'ordinateur à des fins *pédagogiques*, excluant de nombreux points de la description de LANSKY, à savoir les applications purement administratives ou la didactique formelle. Ayant opéré des regroupements parmi les autres points, nous verrons successivement trois grands domaines d'application :

- l'ordinateur comme instrument de laboratoire (Computational Aid)
- l'ordinateur comme instrument de gestion pédagogique (Computer Management)
- l'ordinateur comme instrument d'enseignement (Computer Assisted Instruction)

Une application pouvant, on l'a vu, relever de deux catégories différentes, nous avons fait un choix de classement en fonction de ce qui nous a semblé être sa caractéristique principale.

Le but de la classification qui va suivre est double :

- d'une part, doter le lecteur d'une terminologie bien définie, levant certaines ambiguïtés rencontrées dans le vocabulaire américain et français dans ce domaine, et qui sera utilisée dans la suite du texte,
- d'autre part, décrire un ensemble le plus complet possible des types d'application actuels, ce qui permettra de souligner les points communs de notre travail personnel avec eux, mais aussi son originalité.

Pour chaque type d'application, nous donnerons :

- une présentation la plus simple possible,
- les références de quelques exemples jugés particulièrement originaux.

Puis nous essaierons de dégager :

- les principales caractéristiques pédagogiques et techniques,
- les problèmes qui se posent ; les solutions apportées si elles existent et les actuels thèmes de recherche.

Par caractéristiques pédagogiques, nous entendons les services que l'ordinateur peut rendre à l'étudiant ou au professeur. Ces services sont tantôt présentés en termes des problèmes que les applications peuvent aider à résoudre, au niveau de l'apprentissage, de l'enseignement (selon les définitions données au chapitre 1),

de la motivation des étudiants, tantôt en termes des possibilités nouvelles qu'offrent ces applications au niveau de la transmission et du traitement de l'information.

Les aspects techniques, quant à eux, peuvent être de trois ordres :

- l'équipement et le logiciel requis,
- les contraintes qu'apporte l'application à la technologie éducative,
- la plus ou moins grande facilité d'accès de l'application aux étudiants et aux professeurs.

Notre contribution étant d'ordre méthodologique, nous ne nous intéresserons ni au coût, ni à l'étendue de l'utilisation de l'ordinateur dans l'enseignement.

Les traits communs à plusieurs applications (c'est le cas, par exemple, du traitement du langage naturel) seront examinés lors de la présentation de l'une d'entre elles, mention en étant faite dans la présentation des autres.

2.2. L'ORDINATEUR INSTRUMENT DE LABORATOIRE.

Ici, l'ordinateur n'a pas pour rôle de faire passer de l'information (enseignement), mais il est mis directement à la disposition des utilisateurs (étudiants, professeurs) comme un outil sur lequel ou avec lequel ils peuvent expérimenter. Ce rôle est en grande partie une conséquence de l'importance accordée par la pédagogie contemporaine, nous l'avons vu, à l'activité de l'élève dans la situation d'apprentissage.

Sur le plan pédagogique, l'ordinateur permet alors à l'étudiant de faire exécuter des travaux qu'il a lui-même définis et programmés, ou de faire exécuter des calculs plus ou moins complexes en des temps très courts. Des programmes où l'ordinateur simule des situations expérimentales peuvent permettre à l'étudiant de cerner certains phénomènes à l'étude. Notons qu'ici il est difficile de comparer l'utilisation de l'ordinateur à d'autres techniques puisque celui-ci y joue un rôle original qu'aucune personne ou aucune autre machine ne pouvait jouer.

2.2.1. *L'ordinateur en tant que calculateur.*

L'ordinateur est utilisé comme une machine à calculer, à la différence près qu'il a en mémoire des programmes enregistrés.

Les utilisations sont nombreuses dans les sciences exactes, les sciences humaines (statistiques), et il n'est pas intéressant d'en faire ici une longue liste.

C'est actuellement une des utilisations les plus répandues, surtout si on inclut les micro-ordinateurs qui commencent à se répandre dans les institutions scolaires. Les principaux avantages qu'on lui reconnaît sur le plan pédagogique sont les suivants :

- il permet un premier contact avec l'ordinateur avant même l'apprentissage d'un langage,
- il permet de se concentrer sur les phénomènes étudiés plutôt que sur les calculs à effectuer, évitant des calculs laborieux qui éloignent souvent l'attention des problèmes véritables.

Il n'y a pas d'autres contraintes techniques que celles de la disponibilité du matériel et de sa capacité. L'utilisation peut se faire en traitement par lots ou en mode conversationnel.

Bien que certains auteurs refusent à cet usage de l'ordinateur le statut d'utilisation pédagogique, nous profitons de ce paragraphe pour insister sur le fait que, de notre point de vue, toute application (par exemple l'E.A.O. au sens strict, qu'on étudiera ultérieurement) *doit* permettre à l'étudiant, sous la forme d'une quelconque interruption, d'utiliser l'ordinateur comme calculateur, qu'il soit en train d'apprendre la psychologie, les sciences économiques ou les mathématiques. Nous reviendrons sur ce problème au chapitre 7.

2.2.2. *L'ordinateur objet d'enseignement.*

Il s'agit ici de faire pratiquer aux étudiants la programmation des ordinateurs pour elle-même, soit comme formation professionnelle, soit à cause de la possibilité de transférer les habiletés intellectuelles que cet apprentissage engendre à d'autres activités de l'étudiant [111]. Seule la programmation est envisagée ici, et non les autres enseignements dont l'ordinateur peut être l'objet (technologie par exemple).

Sur le plan pédagogique, cette utilisation a pour caractéristique l'initiative complète de l'étudiant en face de la machine.

L'expérience qui mérite d'être citée ici est celle du langage LOGO, dont les deux fondements sont une recherche sur la didactique des mathématiques et le développement d'un système conversationnel pouvant permettre la programmation facile d'un ordinateur [20, 82, 150, 151].

Les développements auxquels ont donné lieu cette utilisation ont tourné, d'une part autour de la définition de langages d'assez grande puissance bien que simples à apprendre, tels que BASIC, LSE, APL, MATHCALC [180], d'autre part autour de la mise au point de méthodes d'enseignement de la programmation, dont nous parlerons au chapitre 6. Citons également des recherches liées à l'ergonomie [143].

Les similitudes entre la programmation des ordinateurs et leur utilisation en tant que calculateur sont nombreuses, l'étudiant qui n'est pas satisfait des programmes de bibliothèque étant rapidement tenté d'écrire ses propres programmes. Les contraintes techniques sont les mêmes dans les deux types d'utilisation.

2.2.3. *L'ordinateur outil de simulation.*

L'ordinateur sert ici à reproduire un phénomène sur lequel l'étudiant peut expérimenter de façon dynamique.

Les utilisations sont particulièrement nombreuses dans les sciences physiques [73, 113, 126, 207], biologiques et médicales [127] et la gestion (jeux d'entreprises), ainsi qu'en statistiques [75].

Il s'agit d'un type d'apprentissage qui laisse une très large part à la recherche et au travail personnel. L'étudiant qui participe à une simulation n'a pas besoin d'être contrôlé, les effets de ses décisions et de ses actions lui étant visibles à mesure que se poursuit la simulation. Notons cependant, à la lumière des expériences réalisées, que pendant la simulation l'étudiant manque souvent d'explications sur le "comment" des résultats obtenus en fonction des données fournies [44]. C'est un inconvénient que la conception assistée par ordinateur, technique voisine de la simulation dont nous parlerons en 2.4.3., a essayé de pallier.

Sur un autre plan, l'utilisation de l'ordinateur comme simulateur permet d'aborder des expériences ou de présenter des phénomènes avec lesquels l'étudiant ne pourrait pas entrer en contact en temps normal. Une des applications significatives de ce point de vue est l'enseignement du diagnostic médical assisté par ordinateur [54].

Quelles sont les ressources matérielles et logicielles nécessaires ?

La simulation est en général réalisée par un programme écrit en langage évolué et qui utilise soit une loi mathématique, soit un fichier de données. Par voie de conséquence, de la sophistication du matériel utilisé dépendra le raffinement de la simulation, le terminal graphique et un système conversationnel semblant un minimum.

Notons l'intérêt didactique qu'il y a quelquefois à lier l'utilisation de l'ordinateur à la programmation ultérieure du modèle par les étudiants qui l'auront découvert [166].

Signalons pour terminer ce paragraphe la place importante qu'occupe maintenant la simulation en informatique (systèmes d'exploitation des ordinateurs, réseaux d'ordinateurs, systèmes d'information) et qui a débouché sur la définition de langages de simulation (SIMULA...) pouvant maintenant être utilisés dans l'enseignement par la simulation.

2.3. L'ORDINATEUR INSTRUMENT DE GESTION PEDAGOGIQUE.

Dans ce second cas, l'ordinateur, grâce à ses capacités de mémorisation et de calcul, participe à la gestion de l'enseignement (évaluation, allocation de ressources pédagogiques, etc) sans pour autant enseigner directement. Cela correspond à ce qu'on a l'habitude d'appeler l'approche systémique des problèmes d'éducation. Il existe peu d'expériences prenant en compte la globalité de cette approche dans l'"entreprise" universitaire. Citons le système UDEANS décrit dans [117], et dont la figure 5 donne un aperçu. Notons que la plupart de ces systèmes ne sont pas effectivement utilisés, parce que trop lourds, et que souvent ils se contentent de produire des amas de données qui ne sont guère exploitées [44].

Aussi nous n'examinerons que des types *partiels* d'applications : les banques de données, l'utilisation de l'ordinateur dans le processus d'évaluation et enfin la gestion de cheminement, plus connue sous son nom anglais de Computer Managed Instruction (CMI).

2.3.1. *Les banques de données.*

Il s'agit ici des applications où l'ordinateur sert à emmagasiner et à retrouver de l'information. L'utilisateur peut être, selon les cas, le professeur ou l'étudiant. A l'université de Pennsylvanie, par exemple, les étudiants peuvent consulter l'ordinateur pour avoir de l'information sur les professions qui les intéressent. Dans un autre système, les professeurs peuvent produire des listes de questions pour construire rapidement des questionnaires ou des tests.

Les banques de données destinées aux administrateurs ou aux professeurs sont en général accompagnées de programmes de traitement répondant à des spécifications précises, et se comportent souvent comme des systèmes documentaires classiques, avec des langages d'interrogation formalisés. Citons ici l'application décrite dans [34], dont nous aurons l'occasion de reparler dans les chapitres 3 et 7.

Il n'en est pas de même lorsque la banque est principalement destinée aux étudiants, comme la banque de données géographiques du projet SCHOLAR.

Les problèmes qui vont alors se poser sont :

- l'analyse (et éventuellement la generation) du langage naturel,
- la forme sous laquelle l'information doit être rangée,
- la façon de trouver la réponse à la question de l'étudiant.

Nous approfondirons ces différents points dans la suite de ce chapitre, car on les retrouve dans d'autres utilisations. D'ailleurs de telles banques de données, telles que SCHOLAR, sont souvent associées à des systèmes d'interrogation des étudiants.

Nous n'insisterons pas sur l'intérêt des applications "banque de données" à l'époque de l'explosion informationnelle et des progrès réalisés par les télécommunications. En particulier, elles seront un facteur important de développement de ce qu'on a coutume d'appeler maintenant le *travail indépendant*, à l'intérieur ou en dehors de l'institution scolaire.

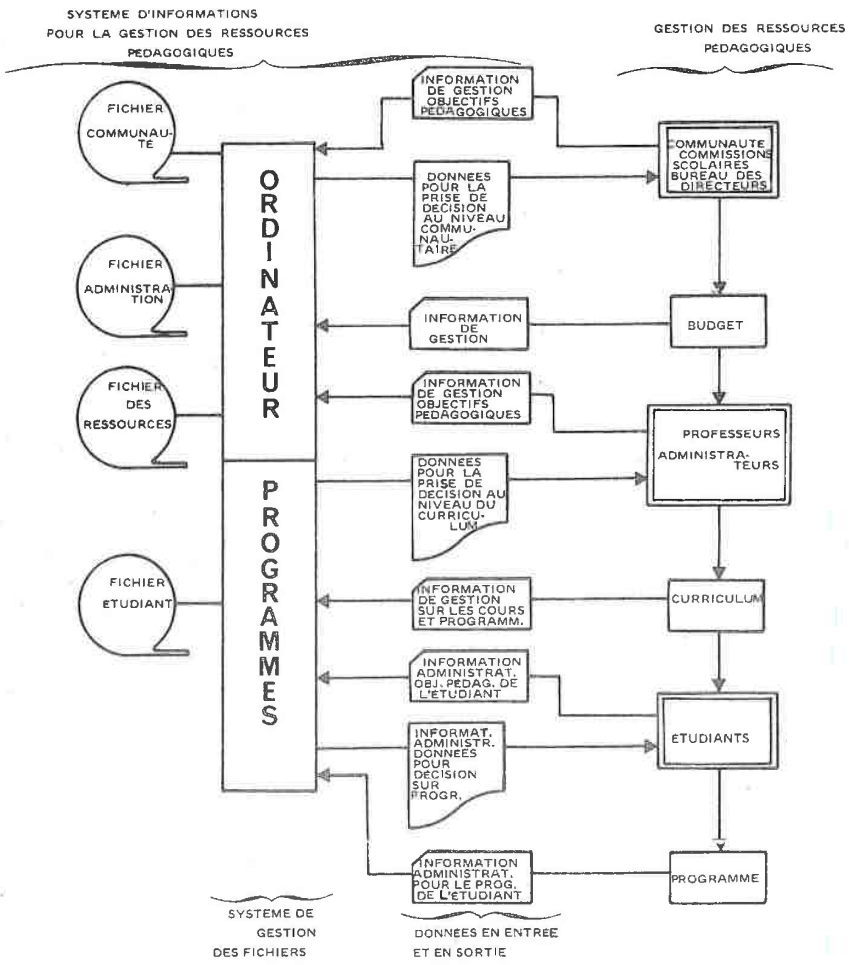


figure 5 : Description du système UDEANS [117]

2.3.2. *L'ordinateur dans le processus d'évaluation.*

Le mot évaluation est à prendre ici dans son sens restrictif d'évaluation des individus ou des groupes. Cependant, outre ce rôle de contrôle de la progression de l'étudiant dans un enseignement individualisé, n'oublions pas le rôle de modification du système global en fonction des résultats obtenus (cf. 1.4.).

2.3.2.1. Analyse et traitement des réponses.

* pour le professeur : l'ordinateur sert à corriger, compiler et analyser les résultats obtenus aux tests par les étudiants, lui fournissant des données statistiques [85].
* pour l'étudiant : en plus de la fonction décrite ci-dessus, possibilité de fournir aux étudiants un corrigé personnalisé, dans le cadre d'un enseignement par correspondance [124, 165, 194].

Cette application nécessite des questions à réponse non construite, en général des Q.C.M. (questions à choix multiples), à la rigueur des réponses numériques.

2.3.2.2. Génération et correction de questionnaires.

Elle n'est, sur le plan informatique, qu'une forme particulière d'utilisation d'une banque de questions, associée à la possibilité pédagogique d'analyse et traitement des réponses décrites ci-dessus. Elle présente au professeur un produit fini utilisable immédiatement dans l'évaluation d'un groupe ou d'un individu.

2.3.2.3. Evaluation individualisée.

Il s'agit d'une génération et d'une correction de questionnaires où chaque question est choisie par le système en fonction des résultats antérieurs de l'étudiant.
L'application nécessite d'avoir formulé les objectifs pédagogiques du cours et de les classer dans une hiérarchie indiquant les prérequis. Il n'est alors plus nécessaire de poser des questions relatives à tous les objectifs, la réussite à une question de niveau supérieur dispensant de l'épreuve des niveaux inférieurs. Chaque étudiant répond, sur un terminal, à un questionnaire différent, taillé à sa mesure [116]. Le problème lié à la forme des réponses sera examiné plus loin. La figure 6, rapportée dans [44], fournit un exemple de hiérarchie de connaissances pour un cours d'algèbre de Boole destiné à des élèves-ingénieurs [105].

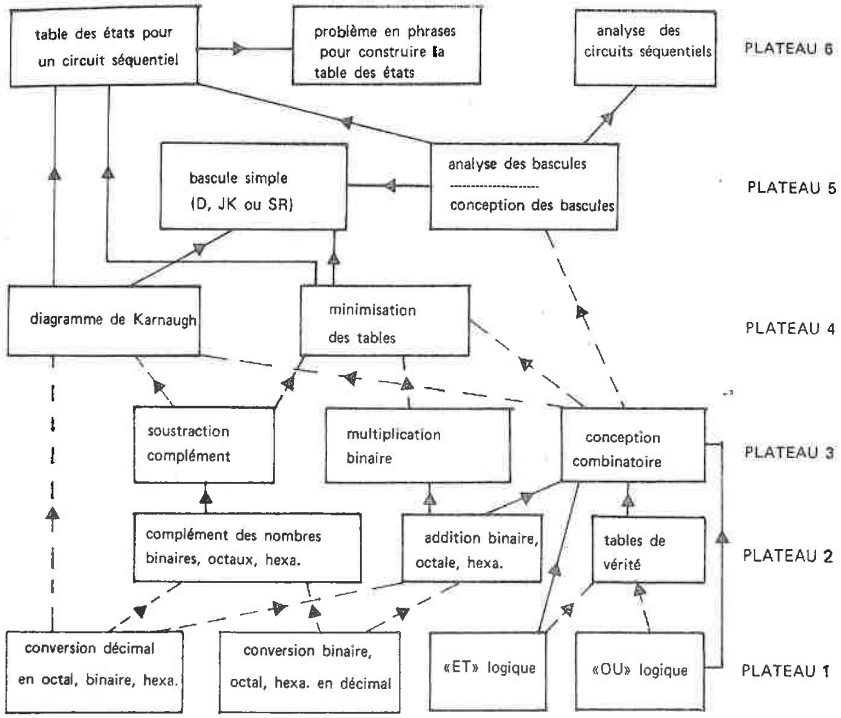


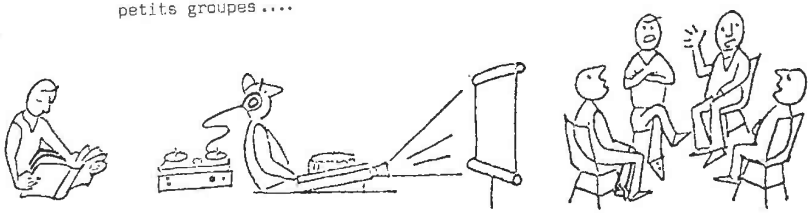
figure 6 : Réseau de connaissances en algèbre de Boole [44]

2.3.3. *La gestion de cheminement.*

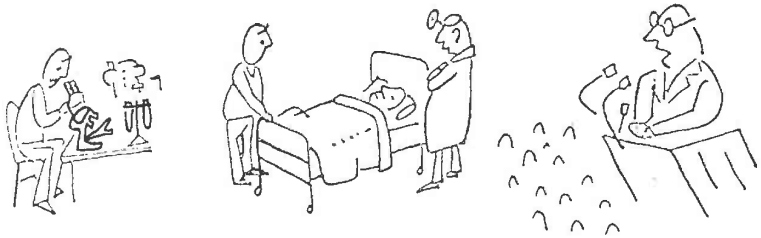
Pour divertir le lecteur, nous allons introduire l'enseignement géré par ordinateur par un petit schéma tiré de [44] ; les deux premières années d'études de l'école de médecine de l'université de Colombus sont organisées de la manière suivante :

. *étape 1* : l'étudiant lit la liste des objectifs pédagogiques à atteindre et des ressources d'enseignement correspondantes.

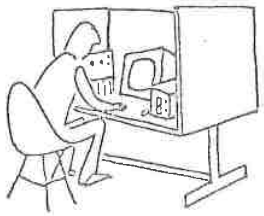
. *étape 2* : l'étudiant choisit quelques-unes de ces ressources en accord avec sa façon de travailler : lecture, moyens audio-visuels, discussion en petits groupes....



travaux pratiques, visites à l'hôpital, conférences.



. *étape 3* : l'étudiant teste ses connaissances sur ordinateur et retourne au besoin à l'étape précédente si les résultats ne sont pas assez satisfaisants.



. *étape 4* : entretien facultatif de l'étudiant avec un enseignant.

. *étape 5* : stage

. *étape 6* : examen écrit et oral

Ce type d'application permet la gestion de l'ensemble des ressources mises à la disposition de l'étudiant et la gestion de la démarche de l'étudiant dans la poursuite des objectifs qu'il s'est fixés. Plus précisément, nous allons donner une liste [2] des fonctions de la gestion de cheminement :

- fonctions standard :

  - stockage des données : des renseignements sur l'étudiant, tels que son nom, son numéro d'identification, les notes obtenues aux tests avec les dates, etc... (cf. 2.3.1.)
  - édition de rapports : élaborés à partir des données stockées (rapports d'évaluation des groupes d'étudiants ou d'individus)
  - évaluation détaillée du cursus (pour évaluer en particulier les matériels pédagogiques)
  - fabrication des tests } voir 2.3.2.2.
  - notation des tests } voir 2.3.2.2.
  - retour des résultats
  - conservation des enregistrements

- fonctions récentes :

  - tests adaptés : procédés très sophistiqués qui règlent les stratégies d'évaluation en temps réel en fonction de la performance de l'étudiant (voir 2.3.2.3.)
  - études à la carte
  - planification et réservation de ressources
  - définition du niveau
  - transmission de messages

Cette application est de plus en plus populaire dans les pays anglo-saxons où l'individualisation de l'enseignement est la plus avancée : PLAN aux Etats-Unis, CAMOL en Grande-Bretagne, le plan KELLER aux Pays-Bas [203]. Quelles sont ses caractéristiques pédagogiques ? L'étudiant a la possibilité d'adapter son rythme de travail à ses performances individuelles. A la limite, c'est lui qui détermine la durée de ses études, et qui choisit le moment où il passera ou repassera un test. En conséquence, le travail principal de l'étudiant ne doit pas être tributaire d'un emploi du temps de cours magistraux. Il s'effectue sur des textes imprimés ou ronéotés, des documents audio-visuels, un terminal d'ordinateur (E.A.O. au sens strict), des documents audio-visuels, et l'étudiant doit pouvoir disposer de ce matériel au moment où cela lui convient individuellement.

La figure 7 donne l'organisation technique d'un système du type PLAN. Ces systèmes utilisent le traitement par lot à distance en dehors des heures normales de classe pour tout ce qui ne nécessite pas de dialogue entre l'étudiant et le système, et le conversationnel pour la partie évaluation des étudiants.

2.4. L'ORDINATEUR INSTRUMENT D'ENSEIGNEMENT.

Ici enfin, l'ordinateur est le support (medium) de l'enseignement, au même titre que peuvent l'être le texte écrit, le professeur en classe, ou le film. A cause de ses caractéristiques propres, l'ordinateur est un medium où peut s'exprimer une interaction entre la machine et l'utilisateur, et où l'enseignement peut s'adapter aux besoins de celui-ci.

Nous aborderons trois types d'applications : les exercices répétés (en anglais "drill and practice"), à propos desquels nous aborderons le problème de l'analyse de la réponse des étudiants ; l'enseignement de type tutoriel, qu'on peut appeler l'enseignement assisté par ordinateur au sens strict, qui est le prolongement direct de l'enseignement programmé issu du behaviorisme (cf. 1.2.), et à propos duquel nous aborderons l'étude de ce qu'on a appelé les langages-auteurs ; les enseignements de type non directif, actuellement encore au stade expérimental, qui

allient aux techniques d'utilisation du langage naturel les techniques dites d'intelligence artificielle.

C'est dans cette dernière rubrique que nous parlerons de la conception assistée par ordinateur, qui utilise elle aussi des techniques d'intelligence artificielle. Ça n'est pas à proprement parler, on le verra, une application pédagogique, mais elle est de plus en plus utilisée dans l'enseignement.

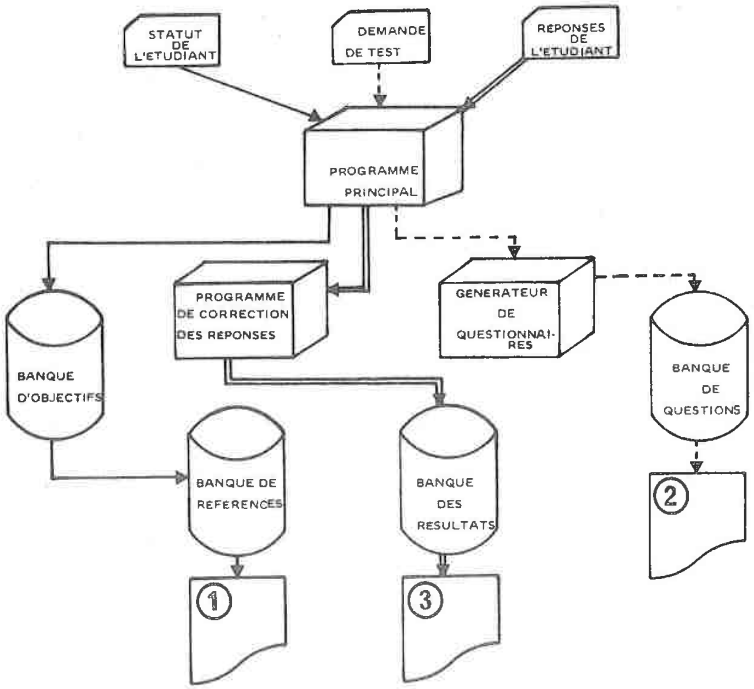


figure 7 : Illustration des composantes d'un système de gestion de cheminement du type PLAN [66]

2.4.1. *Les exercices répétés.*

Il s'agit d'utiliser l'ordinateur comme un répétiteur inlassable donnant à l'étudiant des exercices à résoudre, corrigeant ses réponses et formulant de nouvelles questions aussi longtemps que l'étudiant n'a pas acquis la maîtrise du comportement désiré.

Un des exemples les plus connus est l'ensemble de problèmes recouvrant le cours d'arithmétique du primaire développé par SUPPES à l'université de Stanford [192]. Le stockage des exercices a souvent été remplacé par la génération d'exercices : l'ordinateur fabrique les exercices en choisissant au hasard des valeurs parmi un ensemble spécifié par l'enseignant. Il est doté d'un programme qui lui permet de calculer la solution en prenant pour données les valeurs qu'il a choisies pour l'étudiant. Par exemple, le moule d'exercice :

"J'achète X I1 chez le I2 . Chaque I1 coûte Y francs. Combien dois-je payer ?"

avec les contraintes : $(I1,I2) \in \{(oeuf,crémier),(baguette,boulanger)...\}$

$1 \leq X \leq 100$ et X entier

$0.10 \leq Y \leq 1.50$ et Y décimal à deux décimales

et le but : $Z = X * Y$ francs

permettra d'afficher l'exercice suivant :

"J'achète 3 oeufs chez le crémier. Chaque oeuf coûte 0.60 francs. Combien dois-je payer ?"

et de reconnaître comme réponse correcte "1.80 francs".

Les exercices répétés sont très utilisés en mathématiques et en langues. On trouvera dans [44] la description complète d'un tel système.

Nous allons aborder ici, bien qu'on le retrouve en enseignement de type tutoriel, le problème de l'analyse de la réponse. Que va-t-il se passer si l'étudiant répond :

"2.10 francs" ?

"1.80 franc" ?

"180 centimes" ?

"60 centimes, c'est trop cher, achetez vos oeufs au supermarché" ?

Deux problèmes se posent : comprendre la réponse (c'est-à-dire la reconnaître comme étant correcte ou incorrecte) et envoyer un commentaire approprié. La facilité de compréhension va dépendre de la forme de la réponse [53] : un ou plusieurs caractères (dans le cas des questions à choix multiple), un mot, analysé globalement ou caractère par caractère (problème de l'orthographe), un nombre pur ou avec les unités, une formule, une phrase (suite de mots) d'un langage formel ou naturel, une suite de phrases...

Laissons pour le moment de côté les techniques d'analyse sémantique, utilisant

des modèles de connaissances, et qu'on peut classer en deux rubriques [45] : celles qui se situent dans le cadre de la logique du premier ordre (réseaux sémantiques [209], clauses [40], automates [177]) et celles qui utilisent des modèles moins rigoureux comme MYCIN (inférences [181], grammaires sémantiques [18]), car nous y reviendrons dans la troisième partie de ce paragraphe.

Bornons-nous au problème des mots et, pour les phrases, aux seuls niveaux d'analyse lexicale et syntaxique.

La plupart des systèmes actuellement opérationnels "résolvent" le problème de l'orthographe par l'utilisation de squelettes de mots, et l'analyse lexicale par des fonctions sur des mots-clés. La théorie des grammaires d'ISOSEM de PEUCHOT mérite ici une mention particulière [46]. Il s'agit de construire un modèle de phrase auquel la phrase de l'étudiant sera comparée. La construction a priori par l'enseignant de ce genre de modèle pouvant être difficile, il a été proposé d'utiliser des techniques d'inférence grammaticale pour le faire construire à partir de réponses données par les étudiants et évaluées par le professeur [43, 100].

Citons enfin le langage SNOBOL 4, souvent apparu comme un excellent outil dans la construction d'analyseurs de réponse [11, 118, 148]. On trouvera dans [44] davantage de détails sur les différentes techniques utilisées.

2.4.2. *L'enseignement de type tutoriel.*

L'enseignement programmé, issu de la théorie behavioriste de l'apprentissage, sera approfondi au chapitre 4. Nous n'en indiquons ici que quelques traits utiles pour la compréhension de la suite.

Il consiste à présenter à l'élève, dans un certain ordre, des petites unités de connaissance, puis à vérifier leur acquisition par une question (principe du renforcement immédiat), selon le schéma de la figure 8.

Il était, et est encore, pratiqué avec des livres ou des machines.

En voulant utiliser l'ordinateur, on est tombé une fois de plus, comme le signale BELLISSANT [12], dans le travers courant en informatique qui consiste à refaire avec un ordinateur ce qu'on faisait "à la main" auparavant... l'ordinateur dans ce cas tourne les pages d'un livre, avec ou sans élégance, selon le savoir-faire du réalisateur. Les soviétiques d'ailleurs ne sont pas tombés dans ce travers puisque, n'utilisant pas d'ordinateurs pour des raisons économiques, ils ont réussi à développer des machines de plus en plus sophistiquées, s'appuyant sur des théories non behavioristes de l'apprentissage (algorithmes, apprentissage par résolution de problèmes [193]).

Ceux qui ont développé du logiciel pour ce qu'on a appelé l'enseignement assisté par ordinateur au sens strict ont donc eu à résoudre les trois problèmes suivants :

gestion des items devant être présentés aux élèves, analyse des réponses et détermination du chemin que doivent suivre les élèves dans les différents items (habituellement appelée : stratégie pédagogique) [53].

Ces sortes de logiciels ont reçu le nom de langages d'enseignement, ou encore de *langages-auteur* (i.e. langages permettant aux auteurs d'un cours programmé de l'"entrer" en machine). En 1972, une analyse faite à partir d'une sélection de quatre langages (COURSEWRITER, LA+LDA de l'O.P.E., MAGISTER et PICO) nous permettait d'en dégager quelques standards [175].

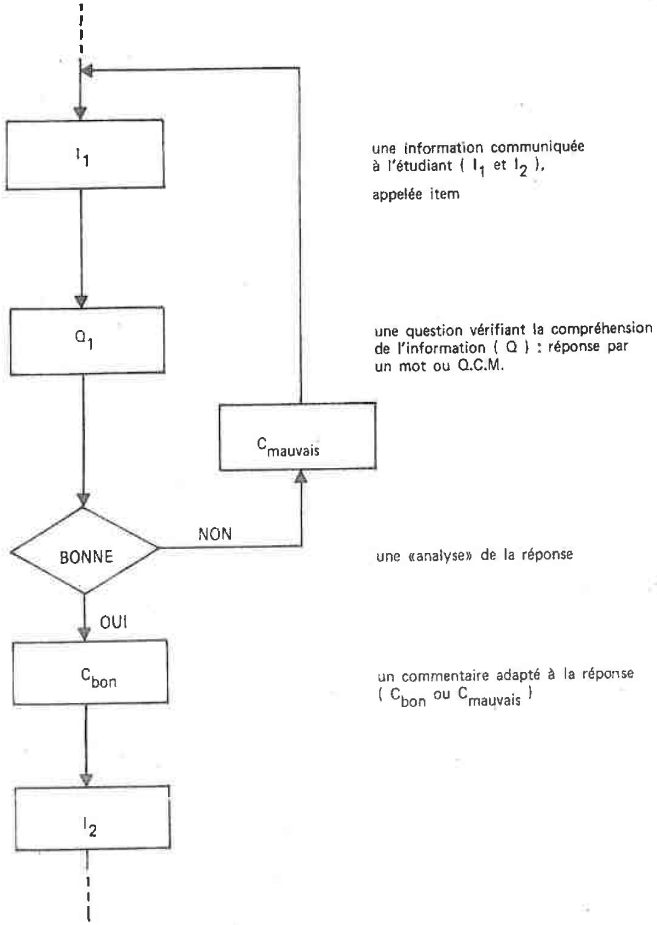


figure 8 : Principe de l'enseignement programmé

Assez rapidement, outre l'analyse de la réponse dont nous avons déjà parlé, les points sur lesquels ont porté les améliorations ont été les suivants : nature des items, élargissement des stratégies d'enseignement, interface du système d'E.A.O. avec les autres ressources de l'ordinateur. Nous allons préciser ces trois points dans la suite. Notons qu'ils font partie de l'inventaire fait par FAFIOTTE en 1975 [59], en même temps que d'autres qui, s'ils ne sont pas détaillés ici, se retrouvent justement dans les propositions de cette thèse :

"Il faut donc ne pas faire l'économie de logiciels offrant des primitives de structuration et de programmation de très haut niveau, dans un formalisme simplifié, logiciels qui devront permettre aux équipes d'enseignants :

- de décrire de manière tout à fait libre
  - . l'organisation logique des activités de l'enseigné,
  - . le contenu informationnel de ces activités,
  - . l'automate d'évaluation ou de contrôle (le plus simple ou le plus adaptatif) qu'ils auront pu élaborer,
  - . les types et les degrés d'initiative laissés à l'enseigné,
- de faire appel à toute ressource logicielle (ou matérielle) externe au programme didactique (petite base d'informations, programmes de calcul numérique, logiciel graphique, activation d'unités audio-visuelles, mini-compilateurs si le dialogue implique une activité de programmation de la part de l'étudiant, etc...) ; l'interface logique entre le logiciel d'E.A.O. et ces ressources ne devra poser aucun problème aux enseignants".

L'ordinateur ne doit en effet pas être un entonnoir, et l'étudiant doit toujours avoir la possibilité de passer à autre chose, en complémentarité ou substitution.

Voyons donc les différentes améliorations apportées aux langages-auteur :

. <u>le contenu informationnel (nature des items)</u>.

Les langages d'enseignement permettent d'écrire sur des fichiers des textes qui constituent le contenu des leçons qu'on va présenter aux élèves. Un problème plus sérieux apparaît lorsque les items écrits sont remplacés par des algorithmes qui seront sous contrôle de l'élève et qui permettront à cet élève de simuler grâce au travail du calculateur des situations réelles. Par exemple, il est possible, par algorithme, de simuler le comportement d'une réaction chimique. Dès lors, l'élève peut à volonté, en modifiant les paramètres de cette réaction, voir se développer devant lui, par le calcul, des courbes qui représentent les états de la réaction qu'il a demandé au calculateur d'effectuer. De telles procédures sont courantes dans le projet PLATO. Dans ce cas, un langage de manipulation de fichiers sera utile, mais bien insuffisant [53]. Il faut un langage qui permette de créer et compiler de telles procédures. Ensuite on doit pouvoir

créer un cours qui les fasse intervenir au même titre qu'un appel de fichiers. Il s'agit donc ici, dans le cas présent, d'intégrer la simulation à l'enseignement de type tutoriel. Dans d'autres disciplines, l'enrichissement des items pourra avoir une autre signification. Nous verrons au chapitre 6 le sens que nous lui donnons dans l'enseignement de la programmation.

. les types et les degrés d'initiative laissés à l'enseigné.

Donner plus de liberté aux étudiants est un des objectifs qu'a poursuivi PEUCHOT en imaginant, à partir du langage COURSEWRITER, la C.M.E.A.O. (conception modulaire de l'enseignement assisté par ordinateur) : l'étudiant peut à tout moment interrompre l'organisation logique de ses activités prévue par le professeur pour accéder lui-même, par des questions, aux informations contenues en machine.

. l'interface avec d'autres ressources.

Il a été partiellement réalisé à l'O.P.E. (mini-langage de programmation et mode machine de bureau) par adjonction de procédures ad hoc.

HEBENSTREIT a dit :

"Si l'enseignement programmé a été progressivement rejeté comme méthode générale d'enseignement, il continue à être partiellement utilisé, et avec succès, dans des domaines où la formation requise s'apparente d'assez près à un ensemble de réflexes conditionnés" [74]. Avec les perfectionnements apportés grâce à l'utilisation de l'ordinateur, et qui permettent de toucher à d'autres types de formation, l'enseignement de type tutoriel reste encore un bon outil d'individualisation de l'apprentissage, d'amélioration de l'enseignement par la rigueur exigée lors de la fabrication d'un cours et la possibilité offerte de le modifier après lecture des renseignements conservés par l'ordinateur sur le comportement des étudiants, et d'alternance en pédagogie. La vitalité d'un centre comme l'O.P.E. en témoigne.

Sur le plan technique, l'enseignement de type tutoriel nécessite un système conversationnel. La taille de l'ordinateur conditionne la longueur et le nombre des séquences pouvant exister simultanément sur le site. On peut utiliser n'importe quel type de terminal. De nombreux centres proposent aux pédagogues des couplages avec des projecteurs de vues fixes ou même des magnétoscopes.

2.4.3. *Les enseignements de type non directif.*

Nous avons déjà examiné des stratégies non-directives d'enseignement : la simulation, les banques de données, les extensions "non-directives" de l'enseignement de

type tutoriel. Dans ce paragraphe, nous voulons aborder les applications utilisant les techniques de l'intelligence artificielle, celles où l'ordinateur se comporte, dans tout ou partie de son activité, comme un "problem solving partner" [154].

2.4.3.1. La conception assistée par ordinateur [33, 179].

La C.A.O. n'est pas, nous l'avons déjà dit en 2.4., de l'E.A.O. Il s'agit d'une technique qui a pour but d'aider l'homme dans les processus de conception. Cette technique est plus particulièrement utilisée dans les bureaux d'études et de recherche pour la mise au point d'ensembles complexes tels que : installations industrielles, bâtiment, architecture, mécanique (conception de machines, pièces à usiner et plus particulièrement réalisation de leur dessin en vue de leur construction ou fabrication), aérodynamique, électronique et électrotechnique (machines électriques, lignes de haute tension), énergie nucléaire, géophysique, informatique...
Technique de conception, la C.A.O. a souvent débouché, particulièrement dans les écoles d'ingénieurs, sur un enseignement par la C.A.O. ; ce type d'enseignement, tout en étant un bon outil d'aide à la formation technique des futurs ingénieurs, confronte l'étudiant à l'utilisation d'une technologie évoluée qu'il sera de plus en plus amené à pratiquer dans son activité professionnelle.
Le système ESPACE [33] nous en fournit un exemple. Il contient deux fonctions principales :

- description par les étudiants de la machine qu'ils conçoivent, ce qui les amène à étudier le problème pour le formuler de manière claire et précise et à approfondir ainsi les relations technologiques qui existent entre les différents éléments de l'ensemble.
- simulation du fonctionnement de la machine conçue, qui met en évidence les relations qui existent entre les grandeurs physiques et les fonctions des diverses parties de la machine.

Les étudiants découvrent ainsi les aspects essentiels de l'élaboration d'un projet industriel, sur les plans :

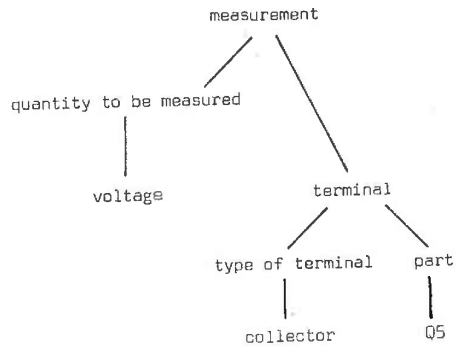
- méthodologique, en effectuant les itérations nécessaires pour satisfaire le cahier des charges.
- algorithmique, en enchaînant des algorithmes pour calculer des parties de la machine ou pour en étudier le fonctionnement.
- heuristique, en combinant, suivant un schéma adapté à leur intuition, leurs connaissances et leur expérience, l'enchaînement d'algorithmes élémentaires mis à leur disposition par le système, pour décrire et simuler leur machine.

Cette application est donc à rapprocher des modes calculateur et simulation. Pour la communication avec la machine, les étudiants utilisent un langage technique proche du français.

2.4.3.2. L'enseignement assisté par ordinateur intelligent.

Sous ce sigle (I.C.A.I. en anglais, ce qui est assez peu flatteur pour leurs prédécesseurs [45]), des chercheurs (principalement américains) ont développé plusieurs projets ayant des finalités assez semblables en chimie [186], trigonométrie [77], géographie [29, 181], biologie [141], électricité [190], logique symbolique [200], théorie des ensembles [187]...
L'idée générale est d'insérer les connaissances d'un expert dans un environnement logiciel capable d'utiliser ces connaissances pour juger l'intérêt des solutions proposées par l'étudiant, l'aider à partir de ce qu'il a déjà fait, le dépanner lorsqu'il ne sait plus où il en est, lui expliquer les raisons du succès ou de l'échec de la méthode qu'il a utilisée,...
Cela suppose que l'ordinateur puisse se mettre dans la même situation que l'étudiant (ignorance de la solution, capacité d'en construire une à partir des données connues de l'étudiant), mais aussi qu'il connaisse les principales erreurs que les étudiants ont l'habitude de commettre, afin de donner une interprétation sensée à d'éventuelles réactions inappropriées.
Le cas le plus simple est celui où le langage de communication est un tant soit peu formel (mathématiques, sous-ensemble du français) et où les "connaissances" se structurent facilement. Dans le cas général, on ne sait actuellement rien sur la façon dont la connaissance est organisée dans le cerveau humain ni sur la façon dont on y accède. On sait simplement qu'elle utilise des dictionnaires, avec des degrés de logicisation variables. On a longtemps voulu croire à des classifications en arbres, mais ceci a été battu en brèche par des expériences récentes. Une organisation en sous-ensembles n'est pas non plus toujours adéquate. Quant à l'accès, par exemple entre les différents sens d'un même morphème, on sait qu'il est direct et guidé par le contexte et non exploratoire (le mot "garde des sceaux" dans une conversation sur la justice n'entraînera pas l'examen de sot, seau ou saut [15]). Par voie de conséquence, les informaticiens ont eu le champ libre pour organiser les connaissances, gérer les dialogues et la résolution de problèmes.
Nous avons déjà cité en 2.4.1. différentes techniques. Nous présentons ici plus en détail l'utilisation des A.T.N. telle qu'elle est décrite dans [27] pour le système SOPHIE. Ce type de formalisation s'applique bien dans le cas d'un domaine limité, donnant lieu à des activités limitées, et où les conceptualisations du domaine sont connues : c'est le cas des circuits électriques.

On appelle grammaire sémantique un moyen simple de tenir compte des pronomalisations, ellipses et autres fragments de phrase qui interviennent naturellement dans une situation de dialogue. C'est ainsi que la phrase "the voltage at the collector of Q5" est engendrée par la structure de contrôle suivante :



La grammaire sémantique est ensuite traduite dans le formalisme des A.T.N. Un A.T.N. (augmented transition network) est un graphe des transitions d'un automate d'état fini, avec les trois différences suivantes :

- adjonction d'un mécanisme récursif qui permet aux étiquettes associées à des arcs d'être des symboles non-terminaux correspondant eux-mêmes à des A.T.N.
- addition sur les arcs de conditions arbitraires devant être satisfaites pour que l'arc soit emprunté.
- addition sur les arcs d'un ensemble d'actions de construction de structure, en même temps qu'un ensemble de registres mémorisant des structures partiellement construites.

C'est ainsi que la règle décrite ci-dessus se retrouve (traits épaissis) dans les trois graphes de la figure 9, dans laquelle CAT MEAS/QUANT est une condition (le mot lu doit être une quantité mesurable), PUSH CIRCUIT/PLACE/ renvoie à un autre graphe par l'intermédiaire du non terminal CIRCUIT/PLACE/. Les actions consistent à mémoriser les mots du vocabulaire terminal qui sont lus. Il existe d'autres actions possibles, par exemple TST pour tester la validité d'une hypothèse qu'on a faite précédemment et mémorisée. La figure 10 donne les composantes et l'organisation du système [94]. Le système de déduction (PROCESS MODULE) est chargé de trouver la réponse à une question, la conséquence d'une proposition de l'étudiant, etc... qu'il envoie à un générateur de phrases. Si la phrase de l'étudiant n'a pas été "comprise", une demande de précision est formulée

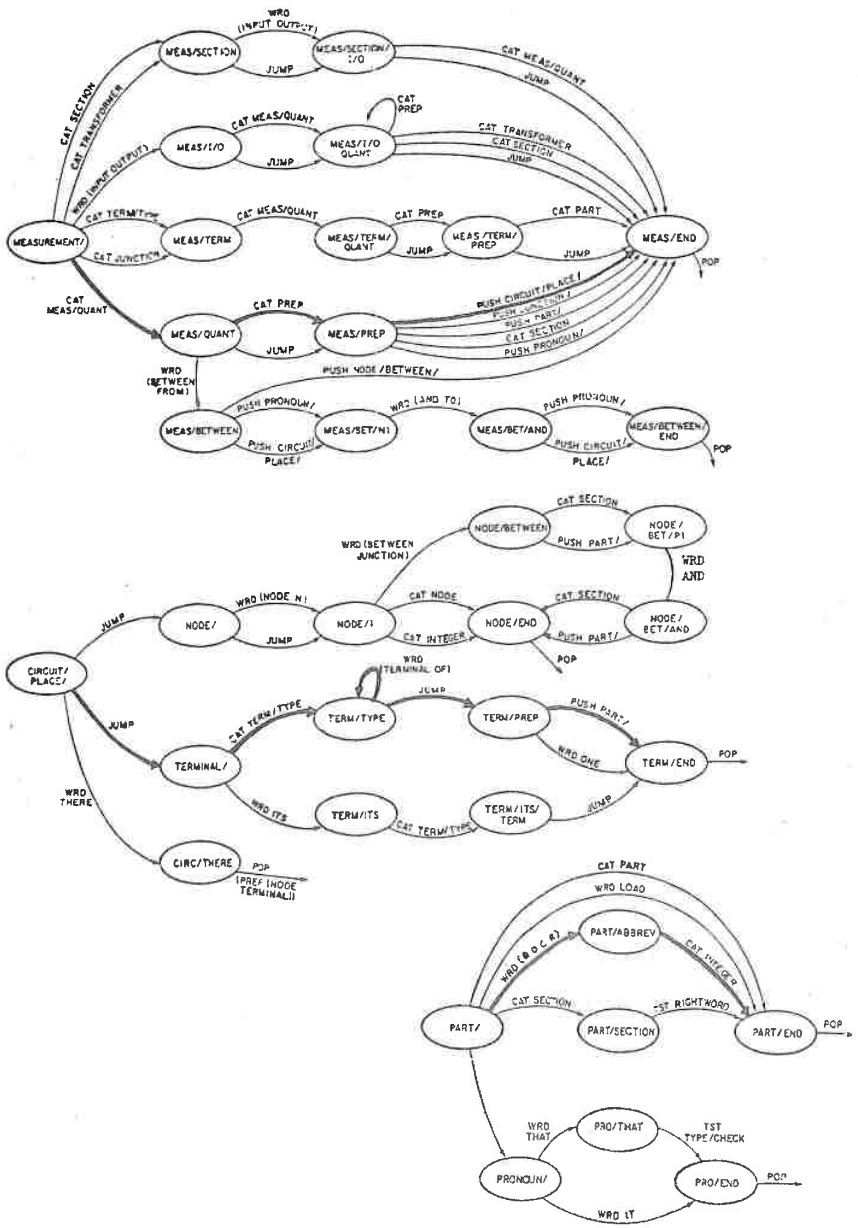


figure 9 : Exemple d'A. T. N. [2]

à partir des hypothèses restées en suspens (par exemple s'il subsiste une ambiguïté à propos d'un pronom).

Les expériences ayant été menées dans des domaines très particuliers, et faute d'éléments de comparaison, on a jusqu'à maintenant mélangé les caractéristiques pédagogiques de ce type d'application et les caractéristiques liées aux domaines. La tendance est actuellement de les séparer.

Les problèmes techniques importants (espace occupé par les programmes d'analyse et les modèles de connaissances, temps d'analyse) font que ces applications relèvent actuellement plus de recherches sur le dialogue homme-machine que de l'utilisation de l'informatique dans l'enseignement.

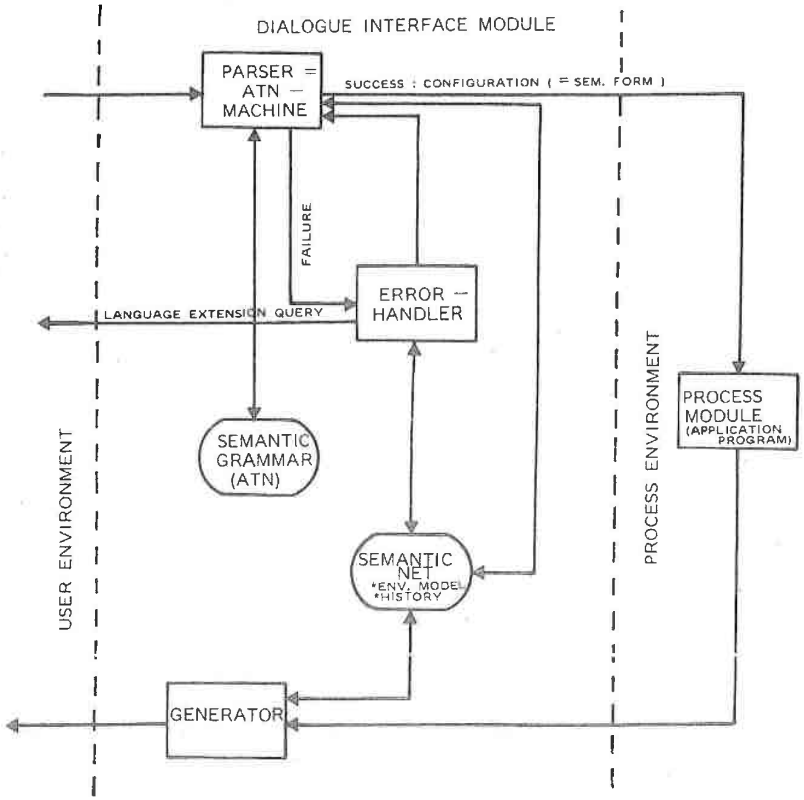


figure 10 : Organisation générale du système SOPHIE [94]

2.5. CONCLUSION.

Cette large palette d'expériences a parfois abouti à la morosité :

"L'E.A.O. au sens strict (en tant que simulation du dialogue entre l'enseignant et l'enseigné) est dans l'impasse... Le bilan actuel des applications de l'intelligence artificielle est assez réduit [50]".

"Former un élève ne se limite pas à communiquer des informations techniques ; aucun robot, si bien programmé soit-il, ne saura prendre à sa charge le colloque singulier de l'enseignant et de l'enseigné [144]".

Mais elle a au moins servi à indiquer des voies de poursuite :

"Le bilan des travaux analysés peut être considéré comme tout à fait positif : même les essais qui n'ont pas abouti ont le mérite de dégager clairement les motifs de ces échecs, et de mieux cerner ainsi les conditions à mettre en oeuvre pour réussir [16]".

"Les applications qui réussissent sont celles qui tentent d'enrichir l'environnement expérientiel de l'étudiant... L'ordinateur favorise l'innovation et la restructuration des programmes d'enseignement, (il permet) de faire de l'"instructional design" [50]".

Que pouvions-nous faire ?

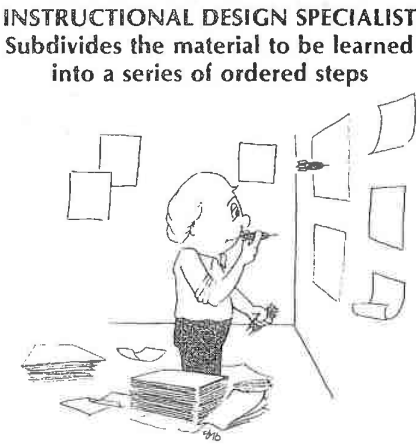
Tout d'abord, nous avons étudié le problème de l'organisation de séquences d'enseignement, dans le cadre général de l'organisation assistée par ordinateur, et nous avons proposé des solutions originales (chapitre 3).

Ensuite, nous avons étudié les modèles d'automate d'évaluation et de contrôle des étudiants, et nous avons proposé un modèle adaptatif valable aussi bien pour la gestion de cheminement que pour l'enseignement de type tutoriel. Dans ce dernier cas, il a été également prévu des possibilités d'interrogation libre par l'étudiant et des fonctions originales d'adaptation de cours et de synthèse (chapitre 4).

Ces deux contributions d'ordre méthodologique ont débouché sur la conception d'un logiciel extensible d'enseignement géré-ou-assisté par ordinateur dans le cadre du projet SATIRE. L'implantation retenue utilise, à des fins d'orthogonalité et de portabilité, un système de gestion de base de données (chapitre 5). La conception est adaptée au système d'exploitation existant et favorise l'utilisation des autres ressources de l'ordinateur (chapitre 7).

La programmation du logiciel a été en partie réalisée (chapitre 8).

Enfin, nous espérons avoir contribué à faire progresser la didactique de la programmation en réalisant pour le projet SATIRE une banque de données et plusieurs cours sur les concepts de base de la programmation. La recherche porte sur le contenu informationnel et l'ordre des activités proposées à l'étudiant, en liaison avec les recherches sur la méthode de programmation déductive [146] (chapitre 6).



CHAPITRE 3 : Analyse et programmation d'un contenu didactique

3.1. INTRODUCTION.

La première partie de ce chapitre a déjà fait l'objet de deux publications [170, 171] dont nous ne reprendrons ici que l'essentiel. Rappelons quelle en était la motivation. Les spécialistes de l'enseignement programmé (cf. 2.4.2. et 4.2.1.) préconisent, avant tout travail de rédaction d'un cours, une analyse rigoureuse de la matière à enseigner, en les différents "atomes" dont chacun fera ensuite l'objet d'un item. Cette analyse, outre son aspect énumératif, doit faire apparaître les relations existant entre les atomes, ces relations étant nécessaires au choix d'un ordre final de présentation des items à l'étudiant. De nombreux auteurs ayant étudié ce problème ont servi de base à notre travail [13, 22, 39, 65, 87, 99, 121, 138, 158, 164, 172].

Mais le problème dépasse largement le cadre de l'enseignement programmé ou de l'enseignement assisté par ordinateur de type tutoriel. On le trouve dans l'organisation d'un cours classique [36, 139] : "un cours étant composé de différents chapitres ayant entre eux des liaisons logiques, il s'agit de trouver un ordre d'exposition tel que le nombre de sauts (i.e. de couples consécutifs de chapitres non liés logiquement) soit minimum. Les problèmes envisagés sont des problèmes d'ordonnancement à contraintes disjonctives". On le retrouve lorsqu'il s'agit de faire la programmation dans le temps d'émissions de télévision éducative dont les sujets ont des rapports entre eux, bien qu'à notre connaissance seuls des moyens empiriques aient été utilisés pour cette question. C'est également un des services pris en charge par l'enseignement géré par ordinateur (cf. 2.3.3.) lorsqu'il aide l'étudiant dans le choix des apprentissages à faire et des ressources associées (voir par exemple [34]). Les exercices répétés l'utilisent également (cf. 2.4.1.).

Il était donc important de présenter le problème dans une certaine généralité : nous nous y employons dans le paragraphe 3.2.

Avant de poursuivre, et de crainte de ne donner plus tard au lecteur qu'une vision ultra simplificatrice de la chose (c'est inhérent à toute formalisation), nous empruntons à VERMERSCH quelques-unes des définitions qu'il a récemment avancées dans ce domaine de la programmation de l'enseignement [201]. Il distingue deux points de vue, et trois aspects, qui sont résumés dans le tableau de la figure 11. L'expert est celui qui est compétent dans une matière donnée, et qui peut a priori répondre à un certain nombre de questions préalables à la programmation de l'enseignement. L'observateur est celui qui recourt de manière systématique à l'analyse et à l'observation de la réalité empirique. Le premier aspect concerne les définitions et relations entre concepts de la matière qu'on veut enseigner. Le second porte sur la séquence des opérations à accomplir pour résoudre une classe de problèmes. Le troisième traite de l'ordonnancement de la progression pédagogique.

	Analyse a priori : POINT DE VUE DE L'«EXPERT»	Analyse a posteriori sur la tâche réelle : POINT DE VUE D'UN OBSERVATEUR
LOGIQUE DES CONCEPTS	BUT : Définir les concepts et leurs relations pour un contenu donné MOYEN : Établissement d'un réseau de relations d'implication, non-linéaire, intemporel ; recherche de cohérence et de systématisation. (analyse de l'«expert»)	BUT : Définir les connaissances effectivement mises en jeu par les sujets en réalisant la tâche MOYEN : Toutes techniques d'objectivation de la «représentation» mises en jeu par un observateur ; étude des verbalisations, des dessins, variations de la tâche ; simulations. . .
LOGIQUE DE L'ACTION	BUT : Définir un algorithme de résolution, c'est-à-dire la séquence des actions nécessaires qui permettra de résoudre une classe de problèmes déterminée MOYEN : Définition, par un «expert», de la classe de problèmes ; définition des variables, opérations et règles ; choix d'une solution algorithmique, en accord avec les finalités que l'on se donne. Relations causales, linéaires, temporelles.	BUT : Définir, par une observation systématique, les actions, décisions effectivement mises en jeu par un sujet effectuant la tâche MOYEN : Observation systématique de sujets qualifiés ; étude de la microgenèse ; procédure de simulation partielle . . .
LOGIQUE PÉDAGOGIQUE	BUT : Définir une progression pédagogique MOYENS : Répartir temporellement la présentation des concepts, le passage d'une activité «théorique» à une activité «pratique», la répétition . . .	BUT : Vérification des résultats au niveau de la compétence et des performances. Valider que ce qui est transmis est bien conforme à l'intention du pédagogue : *validation interne.* Valider l'efficacité par rapport au but de la formation : *validation externe.* MOYENS : Observation du déroulement de la formation, validation empirique des résultats et de la qualité des processus mis en jeu par les sujets

figure 11 : Résumé des principaux aspects de la programmation de l'enseignement [201]

Il est bien évident que le professionnel de la formation, s'il tient compte des deux points de vue, privilégie le point de vue de l'expert. Nous n'y echapperons pas.

Comment maintenant, dans le modèle que nous proposons, se répartissent les trois aspects ? Nous essayons de ne pas faire de distinction a priori entre logique des concepts et logique de l'action : nous appelons l'ensemble *contenu didactique* et nous faisons l'hypothèse qu'il s'agit d'un ensemble fini et d'une ou plusieurs relations sur cet ensemble. Nous appelons analyse du contenu une représentation de l'ensemble et des relations. Pourquoi cette hypothèse ? C'est celle qui nous enferme le moins. Est-elle réaliste ? Nous le pensons en ce qui concerne les mathématiques, sur l'enseignement desquelles nous avons beaucoup travaillé, et nous le démontrons au chapitre 6 en ce qui concerne la programmation informatique. Nous n'en inférons pas la validité sur d'autres disciplines. La logique pédagogique, elle, est la recherche d'un ordre optimal de présentation des éléments de cet ensemble aux personnes en apprentissage (ordre "normal" d'apprentissage, qui ne présage en rien des retours en arrière ou des sauts éventuels de l'étudiant). Plus précisément, il faut déterminer exactement quel est l'élément qui sera présenté après un autre, et qu'on appellera son successeur. Si l'élément α présenté au départ est connu, la suite α, α_1 = successeur (α), α_2 = successeur (α_1)... détermine parfaitement l'ordre de présentation des éléments. La détermination de la fonction successeur est appelée ici programmation du contenu. Notre modèle tient compte des répétitions.

L'objet mathématique jusqu'ici le plus largement utilisé est le graphe ou le multigraphe [14], que ce soit dans l'une ou l'autre application. Par exemple dans [199], "la structure de l'information appartenant à n'importe quel proces d'enseignement programmé est un graphe avec des particularités caractéristiques. Ces particularités sont analysées pour pouvoir traiter le graphe de la façon la plus commode en vue des applications". Et dans [19], "les cours de sciences sont généralement caractérisés par un degré élevé de cohérence, i.e. il existe une dépendance logique entre les concepts individuels qui peut être le mieux décrite comme un graphe orienté à niveau multiple. La séquence des éléments de la matière dans les textes de cours peut être comparée à la structure des graphes. Une procédure visant à créer une séquence d'enseignement optimale sera décrite. Cela implique la découverte du meilleur chemin dans le graphe à niveau multiple orienté et pondéré, constitué par l'ensemble des états de connaissance permis,...".

Enfin dans [34], "pour chaque solution s retenue, la relation $\mathcal{E}$ permet de définir un graphe $\mathcal{G}_s$ orienté des liaisons existant entre tous les modules de cette solution". Pour notre part, étant bien familiarisée avec les graphes qui

font l'objet d'un de nos enseignements, et ayant comparé les deux démarches, nous leur avons préféré les ramifications, objet couramment utilisé dans d'autres domaines informatiques que l'informatique éducationnelle [167]. La programmation sur les ramifications était en effet plus simple que sur un multigraphe, et on verra que les deux représentations sont équivalentes du point de vue de l'analyse. Les ramifications seront rapidement présentées en 3.3.2.2.

3.2. PRESENTATION ET COMPARAISON DES APPLICATIONS.

Le champ d'application d'un modèle d'analyse et de programmation d'un contenu didactique dépend de la définition qui est donnée des éléments de l'ensemble "contenu didactique". La figure 12 indique, pour chaque type d'application, la nature d'un élément, la nature de la ou des relations entre éléments, l'objet mathématique utilisé pour la description, la ou les références bibliographiques concernées.

n°	type d'application	nature des éléments du contenu didactique	nature des relations	objet mathématique utilisé	références bibliographiques
1	enseignement programmé	item = phrase ou groupe de phrases éventuellement accompagné d'aides audio-visuelles	tel item utilise le contenu sémantique de tel autre item, etc...	arborescences, multigraphes ou néant	[3] [22] [39] [121] [172]
2	enseignement assisté par ordinateur de type tutoriel	concepts (terme pris au sens large) mais donnant lieu à un item classique	antériorité, décomposition, équivalence, production, etc...	graphes, multigraphes ou néant	[13] [99] [158]
3	E.A.O. de type tutoriel avec possibilité de stratégie libre	concepts exemple : les modèles de la C.M.E.A.O. (dont la syntaxe est cependant spécifiée)	non précisé	néant	[46]
4	organisation de cursus	modules, chapitres, émissions TV, unités capitalisables, certificats, teaching-learning unit,...	antériorité	graphes	[34] [139] [161]
5	exercices répétés	concepts, parfois répartis en niveaux	antériorité, difficulté	graphes, multigraphes	[105] [192]

figure 12 : Résumé des types d'application mettant en oeuvre une programmation de l'enseignement

L'originalité consiste à proposer un modèle adaptatif, c'est-à-dire dans lequel la nature des éléments et des relations ne sera pas imposée, ni le nombre des relations, ceci étant le propre de la didactique de la discipline considérée ; [170] contient une application à l'enseignement des mathématiques, le chapitre 6 une application à l'enseignement de la programmation.

Il faudra également proposer un modèle efficace, c'est-à-dire dans lequel pourront être prises en compte de manière automatique des contraintes d'ordre didactique, et non l'inverse, à savoir proposer une pédagogie imposée par le modèle. Des exemples de contraintes seront fournis, étant entendu que là encore c'est la didactique de la discipline qui doit fournir ses propres contraintes.

En résumé, les propositions qui vont suivre montrent qu'on peut réconcilier la didactique et l'automatisation de certaines parties du processus d'enseignement. Elles n'ont pas valeur de modèle général puisqu'un choix modifiable a été fait de certains paramètres.

3.3. ANALYSE D'UN CONTENU.

Précisons tout de suite que nous ne suivons pas entièrement dans ce chapitre la classification de VERMERSCH, en ce sens que nous incluons déjà dans l'analyse des éléments de ce qu'il appelle la logique pédagogique. C'est un choix que nous avons fait, de manière à ce que le résultat de l'analyse donne déjà au professeur des indications sur l'utilisation pédagogique qui en sera faite.

3.3.1. *Définition d'un contenu didactique.*

Soit V un ensemble dont les éléments seront appelés des *informations*. Deux sous-ensembles T et N réalisent une partition de V en informations *connues* (ou prérequis) d'une part, et en informations *à apprendre* d'autre part.

Nous supposons que cet ensemble V est muni de relations. Nous donnerons ici deux exemples de relations (un troisième sera fourni au chapitre 6), l'une parce qu'elle se rencontre dans tous les types d'applications cités au paragraphe 3.2., l'autre pour illustrer l'ensemble des relations possibles en enseignement programmé ou en E.A.O. Nous rappelons qu'il s'agit bien ici de connecteurs du second ordre, qui n'excluent pas l'existence de connecteurs du premier ordre *à l'intérieur* des éléments de V, mais que ces derniers ne peuvent être étudiés indépendamment de la discipline.

3.3.1.1. L'antériorité directe $\mathcal{R}$.

$(\forall x \in V,\ y \in N)\ x \mathcal{R} y$ si l'apprentissage de y suppose *directement* la connaissance de x .

$\mathcal{R}$ est antiréflexive, antisymétrique et n'est pas transitive.
On peut définir $\mathcal{R}^p$ pour tout p entier, et on appellera $\mathcal{R}^*$ la fermeture réflexive et transitive de $\mathcal{R}$:

$$(\forall x, y \in V)\ x \mathcal{R}^* y \Leftrightarrow \exists n \in \mathbb{N},\ z_0, z_1 \ldots z_n \in V \text{ tels que } x = z_0 \wedge z_n = y \wedge (\forall i \in \{1,\ldots,n\})\ z_{i-1} \mathcal{R} z_i$$

(l'apprentissage de y implique la connaissance de x par l'intermédiaire de celles de $z_{n-1},\ldots z_1$) .
On notera $\mathcal{R}^+$ la relation d'ordre strict obtenue en prenant $n \neq 0$ dans la définition précédente (fermeture transitive). $\mathcal{R}^+$ est sans circuit.
La relation $\mathcal{R}$ traduit une des formes d'organisation de la matière. Son graphe est sans circuit.

3.3.1.2. La proximité pédagogique $\mathcal{P}$.

$(\forall x, y \in N)\ x \mathcal{P} y$ si l'apprentissage complet (ou une révision complète) de x facilite l'apprentissage de y s'il est fait immédiatement avant, ou après.
La définition laisse croire à une certaine "symétrie" de la relation $\mathcal{P}$, on s'imposera la contrainte :

(1) $\qquad (\forall x, y \in N)$ non $(x \mathcal{P} y$ et $y \mathcal{P} x)$

pour ne pas introduire de circuit dans le graphe, car cela empêcherait la transformation ultérieure du graphe en une structure arborescente.
L'adverbe "immédiatement" introduit dans la définition a pour conséquence que pour tout x de N , il existe au plus un y tel que $y \mathcal{P} x$.
Cependant, si on veut mettre n éléments de N en proximité, on pourra toujours fabriquer une chaîne $x_1, x_2, \ldots, x_n$ avec pour tout $i \leq n$: $x_i \mathcal{P} x_{i+1}$.
Donnons un exemple d'une telle relation dans le domaine de l'enseignement des mathématiques : considérons une information dont le noyau est la notion de relation d'ordre, et une autre dont le noyau est la notion de relation d'équivalence. Il n'existe aucun lien d'antériorité entre ces deux informations. Cependant, on peut estimer intéressant de les enseigner l'une près de l'autre, compte tenu du fait que ce qui les différencie est la propriété d'antisymétrie de l'une qui s'oppose à la propriété de symétrie de l'autre, et que ce facteur peut s'avérer être un facteur de renforcement dans l'apprentissage.

3.3.1.3. Rapports entre $\mathcal{R}$ et $\mathcal{P}$.

$\mathcal{P}$ et $(\mathcal{R}^*)^{-1}$ sont incompatibles. En effet, $\mathcal{R}$ (et donc $\mathcal{R}^*$) traduisent une organisation "logique" de la matière. $\mathcal{P}$ ne doit pas détruire cette organisation

en créant un circuit dans le graphe $(V, \mathcal{R}$ ou $\mathcal{P})$.
On peut avoir $x \mathcal{R} y$ et $x \mathcal{P} y$: cela peut signifier, par exemple, que x est le plus important des prédécesseurs de y pour la compréhension de ce dernier.
Remarquons que les éléments de T n'ont de prédécesseur ni pour $\mathcal{R}$, ni pour $\mathcal{P}$.

3.3.2. *Une analyse d'un contenu didactique.*

Ayant recensé tous les éléments de V et explicité les relations $\mathcal{R}$ et $\mathcal{P}$ sur cet ensemble, nous pouvons les représenter graphiquement par un multigraphe.

Nous appellerons *analyse de contenu* un "découpage" de V en sous-ensembles "complets" d'informations, que nous noterons V_i . Ce découpage ne sera pas comparable à celui d'un magasin en rayons, un ensemble complet d'informations n'étant pas la réunion disjonctive de "tranches de savoir", mais plutôt un ensemble qui grossit chaque fois qu'est présenté un lot d'informations nouvelles. Les V_i seront donc choisis emboîtés $(V_i \subset V_{i+1})$, le dernier étant égal à V . En conséquence, toute différence $V_{i+1} \setminus V_i$ peut, elle, être comparée à un rayon de magasin, ou encore aux "cellules" définies en [172]. Chacune de ces cellules devra correspondre à l'apprentissage d'un *sous-objectif* de l'objectif terminal du cours correspondant à l'ensemble V , c'est-à-dire d'une information qui est le prédécesseur pour la relation $\mathcal{R}$ d'au moins deux éléments de V (sauf les prérequis qui ne sont pas considérés comme des sous-objectifs). On donnera une représentation graphique claire de la restriction des relations $\mathcal{R}$ et $\mathcal{P}$ aux sous-ensembles V_i et aux cellules $V_{i+1} \setminus V_i$.

L'originalité de cette étude, par rapport à celles qui l'ont précédée, est d'avoir accordé de l'importance aux informations qui servent plusieurs fois, et qui étaient génératrices d'ambiguïtés dans les méthodes antérieures [39, 121], sans doute parce que cette situation est très fréquente dans l'enseignement des sciences et des techniques.

La méthode d'analyse de contenu que nous avons choisi de présenter n'est pas, nous l'avons déjà dit, indépendante de la programmation que nous en ferons ultérieurement. Dans cet esprit, certains choix qui peuvent ici paraître arbitraires seront justifiés au paragraphe 3.4. Nous utilisons les relations $\mathcal{R}$ et $\mathcal{P}$, mais n'importe quel pédagogue désireux d'utiliser d'autres relations peut s'inspirer de la méthode pour fabriquer ses propres algorithmes d'analyse et de programmation.

3.3.2.1. Construction d'une ramification $\mathcal{G}$: présentation intuitive.

Il s'agit de transformer le multigraphe $(V, \mathcal{R}, \mathcal{P})$ en une ramification faisant apparaître les V_i et les cellules. L'idée de l'algorithme est la suivante : quand une information est un sous-objectif de l'objectif du cours (au sens défini

plus haut), elle doit se comporter dans un V_i comme l'objectif du cours se comporte dans V (c'est-à-dire apparaître aux racines de la ramification). Cependant on ne doit pas oublier ses relations à ses successeurs (qui seront dans d'autres V_j), donc il existera d'autres occurrences de cette information dans la ramification, qui seront des feuilles, ce qui correspondra à l'idée de révision, alors qu'une occurrence aux racines correspond à l'idée d'apprentissage.

Comment se comporte la relation $\mathcal{P}$? On a dit que si $x \mathcal{P} y$, ce n'est pas seulement x qui facilite l'apprentissage de y, mais sa présentation "complète", c'est-à-dire celle de x et de ses prédécesseurs. Cependant, il faudra bien fixer une limite à cette présentation des prédécesseurs (on ne peut pas "remonter" jusqu'aux prérequis). On trouvera à la figure 13 un exemple d'école où différents cas de rapports entre $\mathcal{R}$ et $\mathcal{P}$ se présentent.

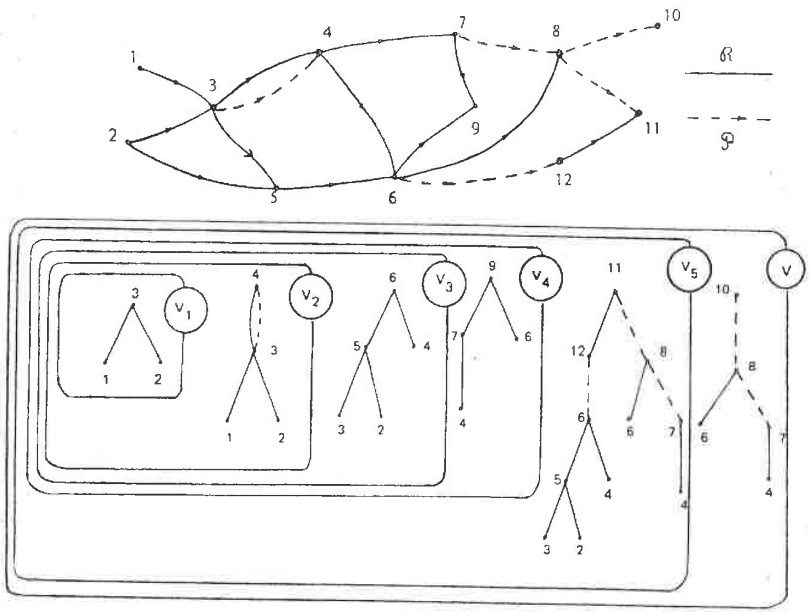


figure 13 : Exemple de graphe et ramification associée

Les objectifs sont les points 9, 10 et 11, les prérequis sont les points 1 et 2. On choisit un objectif terminal (ici 10) et on construit l'arborescence dont il est la racine, en "s'arrêtant" dès qu'on rencontre un sommet qui "sert au moins deux fois" au sens de la relation $\mathcal{R}$ (c'est le cas de 6, 4 et 3). On choisit

ensuite les autres objectifs, et on opère de même. Regardons ce qui se passe pour le sommet 6, qui sert deux fois, mais qui est relié à 12 par $\mathcal{P}$: comme on pense qu'il est bon, avant d'apprendre 12, de bien se rappeler comment 6 a été construit, on ne se contente pas de faire une révision de 6, mais aussi de ses prédécesseurs, en s'arrêtant toutefois à 4 et 3 qui servent eux aussi deux fois. C'est un choix que nous faisons. Cette répétition, au moment de l'analyse, de certains éléments, outre qu'elle fixe immédiatement le souhait du pédagogue, facilitera ultérieurement la programmation.

Quand tous les objectifs ont été pris en compte, on recommence le travail avec les sous-objectifs.

Notons que les prérequis sont, eux aussi, répétés. Comme le note VERMERSCH [201], la logique a priori des concepts présente de nombreux risques, dont celui de dépasser par le bas ce qui est réellement nécessaire à l'élève en n'incluant pas des concepts apparemment très élémentaires pour celui qui analyse la matière, mais dont le rappel est nécessaire pour le niveau des élèves. Il est bon que l'expert en soit très conscient.

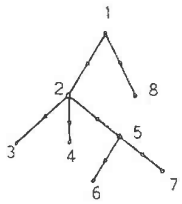
Avant de présenter l'algorithme de façon rigoureuse, nous allons rappeler les quelques définitions sur les ramifications qui nous seront utiles dans la suite. A la présentation algébrique, plus élégante, nous avons préféré la présentation issue de la théorie des graphes qui, on l'a vu, sert de support à la plupart des méthodes d'analyse de contenu.

3.2.2.2. Rappels sur les ramifications.

DEFINITION 1. On appelle arborescence un graphe fini sans circuit (E,Γ) *tel que :*

a) il existe un point v *de* E *qui n'est extrémité d'aucun arc (racine) ;*

b) tout point $x \neq v$ *est l'extrémité d'un arc unique.*



DEFINITION 2. Une <u>*orientation*</u> *d'une arborescence* (E,Γ) *est un ordre partiel* O *dans l'ensemble* E *tel que :*

a) les restrictions de O *à chacun des ensembles* $\Gamma(x)$, $x \in E$, *sont des ordres totaux ;*

b) $\forall y \in \Gamma(x)$ *et* $z \notin \Gamma(x)$, y *et* z *ne sont pas comparables par la relation* O.

Un triplet (E,Γ,O) est appelé *arborescence orientée.*

Soient deux arborescences orientées (E,Γ,O) et (E',Γ',O') ; on appelle *isomorphisme* de la première sur la deuxième une bijection h de E sur E' telle que $(\forall x, y \in E)\ [x\Gamma y \Leftrightarrow h(x)\Gamma' h(y)$ et $xOy \Leftrightarrow h(x)O'h(y)]$ (respect de la relation et de l'orientation)

Etiqueter les points d'une arborescence par des éléments d'un ensemble V revient à définir une application de l'ensemble des points de l'arborescence dans V . Encore faut-il que deux arborescences orientées isomorphes, dont deux points isomorphes ont même étiquette, correspondent à la même structure. Ce qui donne la définition :

DEFINITION 3. Soit V *un ensemble. Dans l'ensemble des couples* (ℓ,f) *formés par une arborescence orientée* ℓ *dont les points sont des entiers, et une application* f *de l'ensemble des points de* ℓ *dans* V *, la relation :*

$(\ell,f) \sim (\ell',f') \Leftrightarrow$ *il existe un isomorphisme* h *de* ℓ *sur* ℓ' *tel que* $f = f' \circ h$

est une relation d'équivalence.

Une *pseudo-arborescence* sur V est une classe de cette équivalence.

DEFINITION 4. Une <u>*ramification*</u> *sur un ensemble* V *est une suite finie de pseudo-arborescences sur* V .

On rappelle que le monoïde libre E^* déduit d'un ensemble E est l'ensemble des suites finies $r_1r_2\ldots r_n$ d'éléments de cet ensemble muni de la loi associative $r_1r_2\ldots r_n + r'_1r'_2\ldots r'_p = r_1r_2\ldots r_nr'_1\ldots r'_p$ et de l'élément neutre "suite qui ne comporte aucun élément" noté Λ .

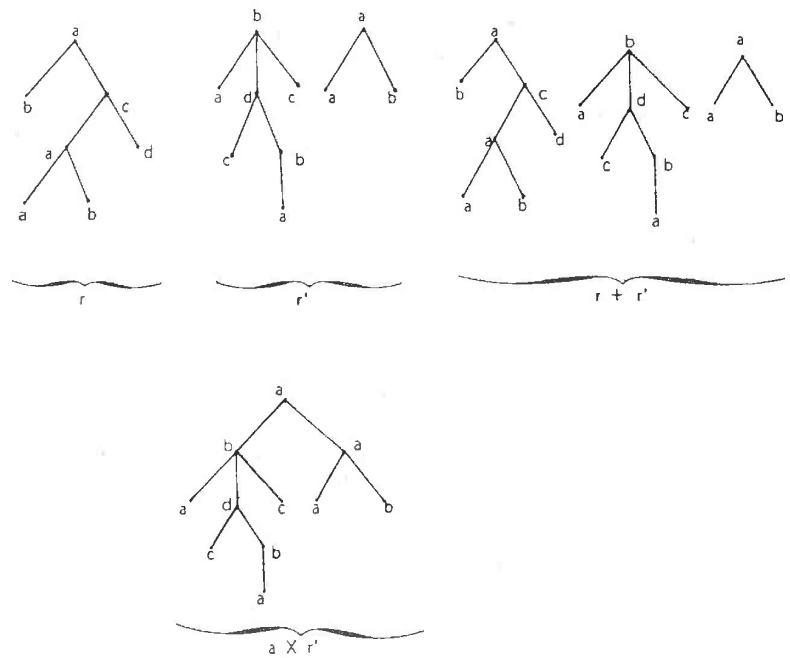
L'ensemble des ramifications sur V sera noté $\hat{V}$; sa loi de composition sera appelée *somme* (figure 14) et notée $+$.

L'élément neutre de cette loi sera noté ramification vide et noté Λ .

DEFINITION 5. On appelle <u>*enracinement*</u> *(figure 14) une loi de composition externe de* $\hat{V}$ *à opérateurs dans* V *, notée* $\times$ *, telle que la ramification* $a \times r$ *soit la pseudo-arborescence obtenue en adjoignant à* r *une racine d'étiquette* a .

Réciproquement, toute pseudo-arborescence s de $\hat{V}$ s'écrit de façon unique sous la forme $a \times r$, $a \in V$, $r \in \hat{V}$.

Il en résulte immédiatement :



<u>*figure 14*</u> : Somme et enracinement dans $\hat{V}$

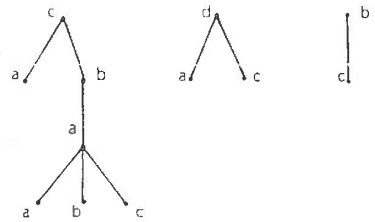
PROPOSITION 1. $\forall r \in \hat{V}$ *, non vide,* $\exists a \in V$, $r' \in \hat{V}$, $r'' \in \hat{V}$ *uniques tels que*

$$r = a \times r' + r'' \ .$$

Et en appliquant plusieurs fois la proposition 1, on obtient :

PROPOSITION 2. Toute ramification non vide sur V *est une combinaison finie par* $+$ *et* $\times$ *d'éléments de* V .

Exemple :



$r = c \times (a \times \Lambda + (b \times a \times (a \times (a \times \Lambda + b + c) + \Lambda) + \Lambda)) + d \times (a \times \Lambda + c) + b \times c + \Lambda$

Le côté agréable de cette définition "algébrique" des ramifications est le principe de récurrence.

PROPOSITION 3. Soit $\mathcal{P}$ un prédicat tel que

a) $\mathcal{P}(\Lambda)$ soit vrai

*b) $(\forall r, s \in \hat{V})$ $(\mathcal{P}(r)$ *et* $\mathcal{P}(s)) \Rightarrow (\forall a \in V)\ \mathcal{P}(a \times r + s)$*

Alors $\mathcal{P}(r)$ est vrai pour tout $r \in \hat{V}$.

Ce principe va nous permettre de définir par récurrence des fonctions de $\hat{V}$ dans un ensemble E.

Mot des racines d'une ramification.

C'est une application $\rho : \hat{V} \to V^*$ (monoïde libre sur V) telle que

a) $\rho(\Lambda) = \Lambda$

b) $\rho(a \times r + s) = a\rho(s)$.

Mot des feuilles d'une ramification.

C'est une application $\varphi : \hat{V} \to V^*$ telle que

a) $\varphi(\Lambda) = \Lambda$

b) $\varphi(a \times r + s) =$ si $r \neq \Lambda$ alors $\varphi(r)\varphi(s)$ sinon $a\varphi(s)$.

Familles d'une ramification.

$\forall a \in V$, c'est une application F_a de $\hat{V}$ dans $\mathcal{P}(V^*)$ ensemble des parties de V^* telle que

a) $F_a(\Lambda) = \emptyset$

b) $F_a(b \times r + s) =$ si $b = a$ alors $F_a(r) \cup \{\rho(r)\} \cup F_a(s)$ sinon $F_a(r) \cup F_a(s)$.

Taille d'une ramification.

C'est une application $t : \hat{V} \to \mathbb{N}$ telle que

a) $t(\Lambda) = 0$

b) $t(a \times r + s) = t(r) + t(s) + 1$

$t(r)$ est notée $|r|$.

Nombre d'occurrences de $a \in V$ dans une ramification.

C'est une application ν_a de $\hat{V} \to \mathbb{N}$ telle que

a) $\nu_a(\Lambda) = 0$

b) $\nu_a(b \times r + s) =$ si $b = a$ alors $\nu(r) + \nu(s) + 1$ sinon $\nu(r) + \nu(s)$.

Branches de racine x d'une ramification.

C'est une application br_x de $\hat{V}$ dans $\mathcal{P}(\hat{V})$ définie par

a) $br_x(\Lambda) = \{\Lambda\}$

b) $br_x(a \times r + s) = br_x(r) \cup br_x(s) \cup \{$si $a = x$ alors $\{a \times r\}$ sinon $\{\Lambda\}\}$.

Hauteur d'une ramification.

C'est une application h de $\hat{V}$ dans $\mathbb{N}$ telle que

a) $h(\Lambda) = 0$

b) $h(a \times r + s) = \max(h(r) + 1, h(s))$.

3.3.2.3. <u>Algorithme de transformation du graphe en ramification</u>.

L'algorithme que nous décrivons est itératif et construit, à chaque étape, un morceau de ramification correspondant à des sous-objectifs. Nous le présentons ici dans une formulation utilisant une procédure récursive.

Soit G un multigraphe $(W, \mathcal{R}, \mathcal{P})$.

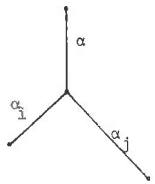
Soit RAM la procédure définie par

RAM(G,s) = <u>si</u> $W \subset T$ <u>alors</u> Λ <u>sinon</u> RAM(G',s') + r

dans laquelle :

- r est une ramification dont les chemins des racines aux feuilles sont les états "maximaux" d'une pile construite sur W et décrite ci-après. Un état maximal u est un élément de W^* tel que si $\alpha \neq \Lambda$, $u\alpha$ n'est pas un état de la pile.

  Deux états maximaux u_i et u_j ayant un facteur gauche commun α donnent la ramification :



$u_i = \alpha\alpha_i$
$u_j = \alpha\alpha_j$
(l'orientation est indifférente)

Description de la pile :

$u_0 = \Lambda$

<u>si</u> $u_i = \Lambda$ <u>alors</u>

  <u>si</u> il existe un point z du graphe appartenant à N, tel que $s(z) = 0$ et n'ayant pas d'entrée inférieure à i, alors i est une entrée de z (entrée d'un objectif non encore pris en compte)

  <u>sinon</u> u_i est le dernier état de la pile

<u>sinon</u> soit x le sommet de u_i

  <u>si</u> $s(x) \leq 1$ <u>alors</u> (x sert au plus une fois au sens de $\mathcal{R}$, on va entrer ses prédécesseurs)

    <u>si</u> il existe y tel que y ($\mathcal{R}$ ou $\mathcal{P}$) x et n'ayant pas d'entrée inférieure à i dans la pile "au-dessus du même mot", <u>alors</u> i est une entrée de y ("au-dessus du même mot" signifie qu'on n'entrera pas deux fois dans la pile un prédécesseur de x s'il n'y a pas eu modification de ce qui est entré avant x dans la pile, problème classique de recherche de chemin dans un graphe)

sinon i est une sortie de x et x est condamné (ceci signifie que x ne servira peut être pas à l'étape suivante)

sinon

si $x\mathcal{P}t$, où t est l'avant dernière lettre de u_i , alors

si il existe y tel que y ($\mathcal{R}$ ou $\mathcal{P}$) x et n'ayant pas d'entrée inférieure à i dans la pile au-dessus du même mot, alors i est une entrée de y

sinon i est une sortie de x

sinon i est une sortie de x (x sera une feuille de l'arborescence, mais est conservé pour une étape ultérieure).

. G' est le graphe obtenu en supprimant dans W les points condamnés qui n'ont pas de successeur ou dont tous les successeurs par ($\mathcal{R}$ ou $\mathcal{P}$)* sont condamnés, et les arcs correspondants.

. s' est la fonction suivante de G' dans $\mathbb{N}$

si ddrp(x) = 0 alors s'(x) = 0 sinon s'(x) = s(x)

où ddrp désigne le demi-degré extérieur de x pour ($\mathcal{R}$ ou $\mathcal{P}$) , ceci afin de se souvenir des "sous-objectifs", même si à cette étape il ne leur reste plus qu'un successeur.

Cet algorithme appelle les remarques suivantes :

- rappelons que le graphe est sans circuit, donc l'algorithme converge (à chaque étape disparaît au moins un point du graphe et ce dernier est fini).
- les points x qui ont au moins deux successeurs pour $\mathcal{R}$ ne disparaissent du graphe que lorsqu'ils ont été appelés dans la pile par chacun de leurs successeurs et que eux-mêmes sont devenus racines de la ramification, grâce à l'artifice qui consiste à conserver leur demi-degré extérieur constant tant que tous leurs successeurs n'ont pas disparu du graphe.
- tous leurs prédécesseurs (au sens de la fermeture transitive) sont protégés de la disparition, bien qu'ils aient pu être condamnés, grâce au mode de construction de G' .

On a alors le résultat suivant :

Soit σ la fonction de $\Gamma = (V,\mathcal{R},\mathcal{P})$ dans $\mathbb{N}$ définie par

si ddrp(x) = 0 alors $\sigma(x) = 0$ sinon $\sigma(x)$ = ddr(x)

où ddr désigne le demi-degré extérieur de x pour $\mathcal{R}$.

Soit $\mathcal{O}$ la ramification RAM(Γ,σ) .

Si $\mathcal{O} = r_1 + r_2 + \ldots + r_p$ dans laquelle les r_i sont des pseudo-arborescences,

$$\begin{cases} \mathcal{O} \\ V_1 = \{x \in V \mid x \text{ a une occurrence dans } r_1\} \\ V_2 = \{x \in V \mid x \text{ a une occurrence dans } r_1 + r_2\} \\ \vdots \\ V_p = V \end{cases}$$

est une analyse du contenu V .

Un programme qui réalise un algorithme du même type, et son exécution sur différents exemples, seront présentés dans les chapitres suivants.

3.3.2.4. Fonctions pédagogiques attachées à l'analyse de contenu.

Replaçons-nous maintenant dans un contexte d'enseignement assisté par ordinateur. Les expériences de REGNIER [172] ont montré que dans le cas d'un enseignement bien organisé, des contrôles effectués non pas après chaque élément de N mais après des "cellules" rendaient l'enseignement programmé plus efficace. Non seulement la transformation du graphe en ramification n'interdit pas de garder la définition des cellules (il suffit alors de donner une définition de fonction de ramification appropriée), mais encore, on peut avoir l'idée d'envisager une planification automatique des fonctions de contrôle et de synthèse, gérée par l'ordinateur de façon conversationnelle avec le pédagogue.

On peut attacher à chaque sommet a de la ramification des fonctions à valeurs entières ou réelles, par exemple :

* $\nu_a(\mathcal{O})$, *nombre d'occurrences de a* dans la ramification, qui signifie intuitivement que a *"sert beaucoup"* pendant l'apprentissage.
* $|\beta(a,\mathcal{O})|$ est le *poids* (nombre de sommets) des branches $\beta(a,\mathcal{O})$ de racine a dans la ramification.

- Envisagée seule, elle peut mesurer la *complexité* de a si on suppose déjà connues les étiquettes des pseudo-arborescences situées à gauche de la première qui contient a dans la ramification.

  Notons que $(\forall a \in T)$ $|\beta(a,\mathcal{O})| = 1$.

  Elle est dans ce cas à rapprocher de la notion de cellule.
- envisagée de façon cumulative, c'est à dire en considérant pour l'ensemble V_a des $x \in V$ tels que x a une occurrence dans $\beta(a,\theta)$ la somme

$$\sum_{b \in V_a} |\beta(b,\mathcal{O})|$$

puis, pour chaque $b \in V_a$, l'ensemble V_b et la somme

$$\sum_{c \in V_b} |\beta(c,\mathcal{O})|$$

et ainsi de suite, elle permet d'avoir une bonne idée de la complexité de a dans l'ensemble du contenu.

Ceci est à rapprocher, quoique différent, du *degré* d'une variable selon PERRIAULT, qui est attachée à la *hauteur* cumulée et non au poids cumulé de la variable [158, 170].

* $i(a)$, *quantité d'information* délivrée par a ; toutes les hypothèses sont permises au pédagogue pour la définition de ce nombre ; nous pensons qu'elle est directement liée à l'objectif du programme. Si le programme veut enseigner une terminologie, c'est à la définition des mots qu'on donnera le plus grand poids. S'il veut apprendre à résoudre des problèmes, c'est sur les propositions au contraire qu'on mettra l'accent. On peut bien sûr cumuler cette quantité d'information, comme dans le paragraphe précédent, en utilisant les familles de prédécesseur a .

Décider de faire un contrôle ou une synthèse après la présentation de a , et quelle sorte de contrôle ou de synthèse, peut dépendre de la valeur prise par ces fonctions :

"$i(a) \geq$ seuil fixé" peut vouloir dire que a est un sous-objectif dont il faut vérifier l'acquisition,
"$v_a(\mathcal{O}) \geq$ seuil fixé" impose de bien s'assurer de l'acquisition de a , qui est une information clé de la discipline,
"$|\beta(a,\mathcal{O})|$ simple ou cumulée $\geq$ seuil fixé" impose de poser des questions "intelligentes" sur les constituants de a , nombreux à être mis en relation,
"$i(a)$ cumulée $\geq$ seuil fixé" peut signifier qu'il est temps de faire le point.

Ainsi toutes les fonctions paraissant intéressantes à un pédagogue pourront être prises en compte dans un système conversationnel d'aide à l'analyse et à la programmation d'un contenu didactique, l'algorithme effectuant les calculs des valeurs prises par certaines de ces fonctions sur la ramification, ou interrogeant le professeur sur la valeur des autres.

3.4. PROGRAMMATION D'UN CONTENU.

3.4.1. *Définition.*

Nous appelons programmation d'un contenu V ce que VERMERSCH appelait la logique pédagogique, à savoir la définition d'un ordre de présentation des éléments de V tenant compte des contraintes de logique de la matière et logique de l'action représentées par les relations existant sur V .

En ce qui nous concerne, prenant en compte les relations $\mathcal{R}$ et $\mathcal{P}$ du paragraphe 3.3., nous fabriquerons un mot β de V^* satisfaisant aux contraintes hiérarchisées (1) à (6) développées ci-après :

(1) Pour tout i , il existe un facteur gauche β_i de β qui soit une programmation du contenu de V_i (respect de la modularité de l'analyse).

(2) $\forall a \in V$, $s_k(a) = p \Rightarrow a$ apparaît au moins p fois dans β , ce qui signifie que toutes les informations de V sont présentées, et qu'elles sont répétées autant de fois que nécessaire. Si $s_k(a) = 0$, le nombre d'occurrences de a dans β est égal à 1 .

Dans la suite, on notera x_0 la première occurrence de x dans β ; elle sera appelée *présentation initiale* de x , les autres seront appelées *répétitions*.

La contrainte (2) mérite quelques explications. Elle ne s'applique pas de la même façon aux applications recensées en 3.2. Elle n'a en particulier aucun sens en enseignement programmé, dont les atomes ont un contenu sémantique très réduit, et peu de sens en E.A.O. de type tutoriel classique. Elle trouve sa place dans les applications 3 à 5, étant entendu qu'une répétition n'est pas une reproduction identique à la présentation initiale, mais un simple rappel des atomes qu'il faut connaître avant d'aborder un nouvel atome, ou qui sont utiles d'un autre point de vue. Donc, selon les cas, cette contrainte sera prise ou non en considération, ce qui pourra conduire à des modifications dans l'algorithme qui sera présenté en 3.4.3.

On appellera dans la suite *séquence* (définition inspirée par [39]) tout facteur α de β de longueur maximale et tel que

$(\forall x, y \in V)\ (\forall \delta, \gamma \in V^*)\quad \alpha = \delta x y \gamma \Rightarrow x\ (\mathcal{R} \text{ ou } \mathcal{P})\ y$,

ce qui signifie qu'à l'intérieur d'une séquence deux lettres consécutives représentent des informations en relation, et que la séquence est rompue quand il n'y a plus de relation.

On appellera *présentation complète* de x dans β tout facteur S_x de β dont la dernière lettre est x (ou x_0) , de longueur maximale et tel que

$(\forall a \in S_x)\quad a\ (\mathcal{R} \text{ ou } \mathcal{P})^+\ x$.

Notons que $x \in T \Rightarrow S_x = x$.

Les deux contraintes qui suivent précisent le rôle des relations $\mathcal{R}$ et $\mathcal{P}$ dans la programmation du contenu :

(3) $x\mathcal{R}y \Rightarrow x_0$ est à gauche de y_0 dans β
(par transitivité, ce résultat s'applique à $\mathcal{R}^*$)

(6) $x\mathcal{P}y \Rightarrow y_0$ est directement précédée (cas $x\mathcal{R}y$) ou suivie (cas $\neg x\mathcal{R}y$) d'une présentation complète de x , initiale ou répétée. Notons qu'avec le travail préparatoire que nous avons fait dans l'analyse du contenu, il suffit de dire que y_0 est directement précédé ou suivi de x .

Enfin, dans le cas où on ne travaille qu'avec la seule relation $\mathcal{R}$, les deux contraintes suivantes doivent être satisfaites dans cet ordre :

(4) Le nombre de séquences de β est minimal ; intuitivement, un changement de séquence correspond à une rupture momentanée de l'ordre logique, et on souhaite qu'il y ait le plus petit nombre possible de ces ruptures.

(5) Si on appelle $d(x,y)$ le nombre des informations qui séparent les deux occurrences les plus proches de x et y dans β , la quantité, calculée pour y fixé,

$$\sum_{\{x \mid x\mathcal{R}y\}} d(x,y_o)$$

est minimale, ce qui signifie que les antécédents directs de y doivent, dans leur ensemble, être présentés le plus près possible de y .

3.4.2. *Idée intuitive de l'algorithme.*

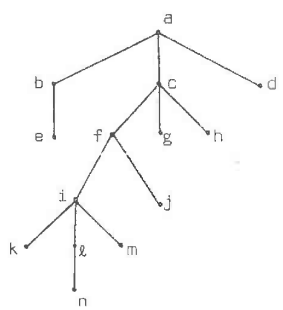
Pour programmer le contenu de V , il suffit de "lire" les étiquettes de la ramification $\mathcal{O}$ en respectant les contraintes (1) à (6) . La condition (2) est alors automatiquement vérifiée.

La condition (1) sera vérifiée pourvu qu'on lise les pseudo-arborescences de $\mathcal{O}$ de gauche à droite.

La condition (3) également, si on impose en plus de lire chaque pseudo-arborescence r_i de bas en haut, c'est-à-dire en ne lisant une étiquette a qu'après avoir lu les informations de $F_a(r_i)$.

Il nous reste donc à choisir un algorithme de lecture tel que le résultat obtenu vérifie les contraintes (4) à (6) .

Commençons par le cas un peu particulier mais plus contraint où on ne travaille qu'avec la seule relation $\mathcal{R}$, et prenons un exemple :

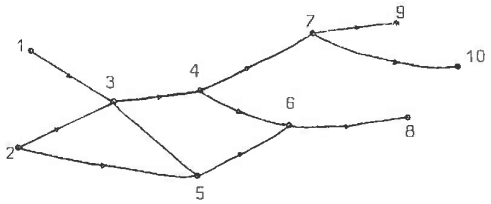


Supposons que l'étiquette de la dernière racine lue soit e . Si on commence la lecture de la pseudo-arborescence par e :

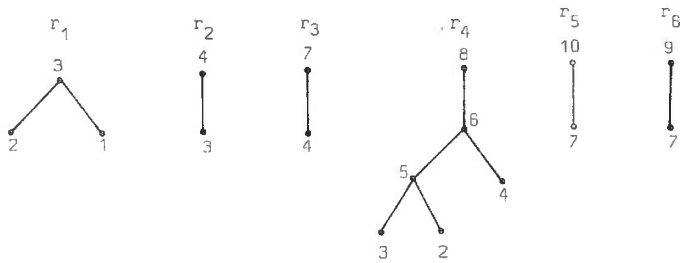
- On n'a pas besoin de la répéter, car il figure dans la ramification au moins $s_k(e)$ fois plus une, et donc sans répétition son nombre d'occurrences dans β sera suffisant.
- On peut lire ici b à la suite, en poursuivant la séquence commencée (économie d'une séquence).

Il semble donc que ce soit un bon choix pour satisfaire la condition (4), à condition que la racine de r_i apparaisse toujours aux feuilles de r_{i+1} . Ce n'est pas toujours le cas.

Considérons en effet le graphe

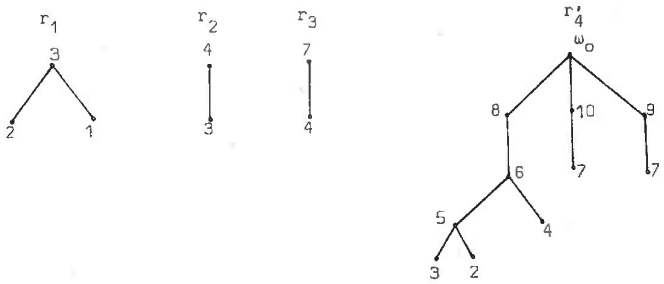


En fonction de l'ordre d'entrée dans la pile, lors d'une étape de la construction, des points de demi-degré extérieur nul, on obtiendra par exemple la ramification $\mathcal{O} = r_1 + r_2 + r_3 + r_4 + r_5 + r_6$



ou la ramification $\mathcal{O}' = r_1 + r_2 + r_3 + r_5 + r_4 + r_6$. D'un certain point de vue, il y a "commutativité" de l'addition sur r_4, r_5 et r_6 et le choix de $\mathcal{O}$ "coûte" une séquence de plus que celui de $\mathcal{O}'$.

Dans ce cas, pour supprimer l'ordre fictif entre r_4, r_5 et r_6, il suffit d'enraciner ces trois pseudo-arborescences à une étiquette "de service" ω_o ne faisant pas partie de V . La ramification de l'exemple devient :



En pratique, et puisque cette anomalie provient de l'algorithme de construction de la ramification, on aura intérêt, plutôt que rechercher un algorithme de regroupement des pseudo-arborescences "commutatives", à modifier l'algorithme de construction du paragraphe 3.3.2.3. de la façon suivante :

. G' est le graphe obtenu..... correspondants. Si le nombre des sommets a_i de G' de demi-degré extérieur nul pour $\mathcal{R}$ ou $\mathcal{P}$ est supérieur ou égal à deux, ajouter au graphe le sommet ω_o et les arcs $a_i \overset{\mathcal{R}}{\leftarrow} \omega_o$ correspondants.

On a alors les résultats suivants :

(R1) A des permutations près sur les branches de racine a, $a \in V \cup \{\omega_o\}$, la ramification obtenue est unique.

(R2) Soit $\mathcal{G} = r_1 + r_2 + \ldots + r_n$; pour tout $i = 1 \ldots n-1$:

1) si la racine de r_i est différente de ω_o, elle apparaît aux feuilles de r_{i+1}.

2) si $r_i = \omega_o \times (r'_1 + \ldots + r'_p)$, pour tout $j = 1 \ldots p$, la racine de r'_j apparaît aux feuilles de r_{i+1}.

Revenons à la contrainte (4) ; pour la satisfaire, on s'efforcera de construire les séquences les plus longues possibles (puisqu'à nombre de lettres égal, plus les mots sont longs, moins ils sont nombreux), en "remontant" les chemins des feuilles à la racine en ne s'arrêtant qu'aux étiquettes dont on n'a pas encore lu tous les antécédents.

En ce qui concerne la condition (5), elle sera satisfaite en lisant les branches de racine a, pour lesquelles l'ordre est indifférent pour (4), dans l'ordre des tailles décroissantes. En cas de tailles égales, on lira de gauche à droite.

Sur notre exemple, la programmation du contenu sera

$$\beta = \underset{\alpha_1}{2} \,|\, \underset{\alpha_2}{1\ 3\ 4\ 7\ 10} \,|\, \underset{\alpha_3}{3} \,|\, \underset{\alpha_4}{2\ 5} \,|\, \underset{\alpha_5}{4\ 6\ 8} \,|\, \underset{\alpha_6}{7\ 9} \,|$$ (ω_o n'est pas "lu")

Il restera uniquement à régler le cas d'occurrences multiples de a aux feuilles.

Il est trivial que la présence de la relation $\mathcal{P}$ vient perturber ce processus d'optimisation, par création de séquences supplémentaires, et augmentation de la distance de a à ses prédécesseurs.

Cherchons en effet les incidences de la contrainte (6). Soit x une étiquette de r_i. Ce qu'on a appelé présentation complète de x est exactement le résultat de la branche de racine x dans la ramification r_i. Pour satisfaire la contrainte (6), il suffira de lire cette branche en dernier, juste avant de lire y (cas $x\mathcal{R}y$) ou après (cas $\neg x\mathcal{R}y$: gain d'une séquence).

Si on reprend maintenant la ramification du paragraphe 3.3.2.1., on obtient la programmation suivante (9, 10 et 11 sont enracinés à ω_o) :

$$\beta = 1 \,|\, 23 \,|\, \underbrace{1}_{\text{répétition}} \,|\, 234 \,|\, 3 \,|\, 256 \,|\, 12 \,|\, \underbrace{3}_{\text{répétition}} \,|\, 25 \,|\, 46 \,|\, 11 \,|\, 68 \,|\, 47 \,|\, 10 \,|\, \underbrace{68}_{\text{répétition}} \,|\, \underbrace{47}_{\text{répétition}} \,|\, 47 \,|\, 69$$

dans laquelle figurent de nombreuses répétitions (mais c'est un exemple technique et non didactique).

3.4.3. *Algorithme de programmation.*

Il est défini par récurrence grâce à deux fonctions

$der(r,x) : \hat{V} \times V \to V$ qui désigne la dernière lettre lue et qui est initialisée à $* \notin V$.

$lec(r,x) : \hat{V} \times V \to V^*$ qui décrit le résultat.

(a) $lec(\Lambda,x) = \Lambda$
$der(\Lambda,x) = x$

(b) $lec(x,x) = \Lambda$ cas des feuilles : pas de répétition immédiate (on gagne une séquence)
$der(x,x) = x$

(c) $lec(r + r',x) = lec(r,x)lec(r',der(r,x))$
$der(r + r',x) = der(r',der(r,x))$

(d) si $r \neq \Lambda$ et $r = \sum_{i=1}^{n} r_i$ où les r_i sont des pseudo-arborescences

$$lec(a \times r,x) = lec\left(r_k + \sum_{i=1}^{p} r_{j_i}, x\right)\alpha$$ dans lequel

- r_ℓ est, parmi les r_i, celle qui a une racine b telle que $b\mathcal{P}a$, ou bien Λ.
- r_k est, parmi les $r_i \neq r_\ell$ et ayant x aux feuilles, celle qui a la plus grande taille, ou bien Λ.

- $r_{j_1},\dots r_{j_p}$ sont les $r_i \neq r_k$ et r_ℓ et rangées par tailles décroissantes.
- α et $der(a \times r,x)$ sont définis de la façon suivante :

. $a \neq \omega_o$ (dans les deux cas, $r_\ell = \Lambda \Rightarrow \alpha = a$ et $der(a \times r,x) = a$)

<u>si</u> $b\mathcal{R}a$ <u>alors</u> $\alpha = lec(r_\ell, der(r_{j_p},x))a$; $der(a \times r,x) = a$

<u>sinon</u> $\alpha = a\ lec(r_\ell,a)$; $der(a \times r,x) = der(r_\ell,x)$

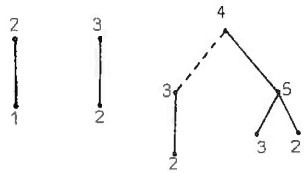
. $a = \omega_o$ (dans ce cas, $r_\ell = \Lambda$)

$\alpha = \Lambda$

$der(\omega_o \times r,x) = der(r_{j_p},x)$

3.4.4. *Vérification des contraintes (4), (5) et (6).*

Nous avons plus haut parlé de contraintes hiérarchisées dans le sens où, si elles sont incompatibles, nous nous attacherons à respecter d'abord la première, et à nous contenter de compromis dans la suite. Nous pouvons tout de suite noter que le résultat de cet algorithme ne vérifie pas la condition (6). On le voit sur ce contre-exemple



qui fournit le mot $\beta = 1\ 2\ 3 \mid 2\ 5\ 4 \mid 2\ 3$ dans lequel $1\ 2\ 3$ est la présentation complète de 3 alors que 4 est suivi uniquement de $2\ 3$. Ceci provient de la recherche du minimum de séquences par (b) et la recherche de r_k pour vérifier (4). En effet, la suppression de (b) implique ici

$$\beta = 1\ 2 \mid 2\ 3 \mid 3 \mid 2\ 5\ 4 \mid 2\ 3 \mid$$

et on pourrait démontrer que la condition (6) est vérifiée.

On ne considère plus maintenant que l'ensemble des ramifications obtenues à partir de la seule relation $\mathcal{R}$, et on ne tient pas compte des racines fictives ω_o, dont le but est de se prémunir contre des séquences inutiles (résultat (R2)) mais qui disparaît à la lecture.

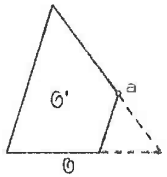
Si on note $ns(\mathcal{O})$ le nombre de séquences de $lec(\mathcal{O},y)$ où y est l'élément approprié de V, on a les résultats suivants :

(R3) $\mathcal{O} = a \in T \Rightarrow ns(\mathcal{O}) = 1$

$\mathcal{O} = \mathcal{O}_1 + \mathcal{O}_2 \Rightarrow ns(\mathcal{O}) \leq ns(\mathcal{O}_1) + ns(\mathcal{O}_2)$

$\mathcal{O} = a \times \mathcal{O}' \Rightarrow ns(\mathcal{O}) = ns(\mathcal{O}')$

(R4) Soit $\mathcal{O}'$ la ramification obtenue en remplaçant la branche de racine $a \notin T$ dans $\mathcal{O}$ non réduite à une feuille par la feuille a.



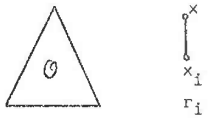
$$ns(\mathcal{O}') = ns(\mathcal{O}) - ns(br_a(\mathcal{O})) + 1$$

Nous allons maintenant regarder comment évolue $\mathcal{O}$, et par voie de conséquence $lec(\mathcal{O},*)$, quand on ajoute un élément à V.

Notons tout d'abord que cet élément x est nécessairement un objectif terminal de $V' = V \cup \{x\}$, car tous les prérequis et objectifs intermédiaires ont été recensés dans V.

Soient $x_1, x_2, \dots x_n$ les éléments de V tels que $x_i \mathcal{R} x$.

(a) Si $x_i \in T$, $\mathcal{O}'$ se déduit de $\mathcal{O}$ par adjonction de la pseudo-arborescence r_i



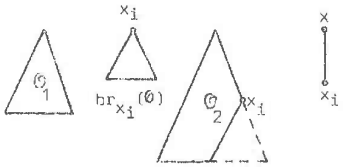
et $lec(\mathcal{O}',*)$ comporte une séquence de plus que $lec(\mathcal{O},*)$.

(b) Si $x_i \notin T$ et n'est pas objectif terminal de V

(b_1) si x_i apparaît aux feuilles de $\mathcal{O}$, $\mathcal{O}'$ se déduit de $\mathcal{O}$ par la même transformation qu'en (a).

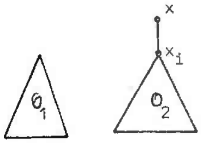
(b_2) si x_i n'apparaît pas aux feuilles de $\mathcal{O}$, $\mathcal{O}'$ se déduit de $\mathcal{O}$ par :

- adjonction de r_i
- remplacement dans $\mathcal{O}$ de la branche de racine x_i par x_i
- adjonction de cette branche en tant que pseudo-arborescence



dans ce cas, $lec(\mathcal{O}',*)$ comporte deux séquences de plus que $lec(\mathcal{O},*)$.

(c) Si x_i est un objectif terminal de V, x est enraciné à la pseudo-arborescence de racine x_i dans $\mathcal{O}$



et $lec(\mathcal{O}',*)$ comporte le même nombre de séquences que $lec(\mathcal{O},*)$.

Supposons que les prédécesseurs de x se répartissent en n_1 éléments de type (a), n_2 de type (b_1), n_3 de type (b_2) et n_4 de type (c). Les transformations décrites ci-dessus s'appliquent sans interférence et $lec(\mathcal{O}',*)$ comporte $n_1 + n_2 + 2n_3$ séquences de plus que les $lec(\mathcal{O},*)$. On ne peut pas faire mieux à cause de la condition (2) qui oblige à répéter une information dès qu'elle sert deux fois. Nous venons donc en quelque sorte de démontrer par récurrence sur le nombre d'éléments de V le résultat

(R_5) Le nombre de séquences de β est minimal.

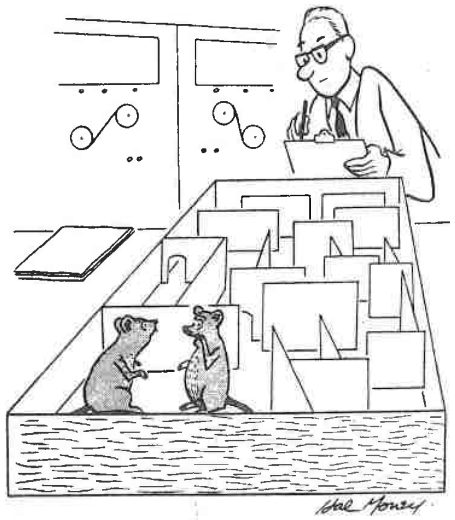
La contrainte (6) est d'un examen plus délicat. Que faut-il minimiser ? On suppose a priori que l'analyse de contenu a été faite sous la forme de la ramification $\mathcal{O}$, et non sous une autre forme, car dans ce cas toute démonstration est impossible. L'idée naive consistant à lire les antécédents de y juste avant y ne tient pas, car elle ne s'exprime pas de manière récursive. Soit donc r la branche de racine y dans $\mathcal{O}$, non réduite à une feuille.

$$r = y \times (r_1 + r_2 + \ldots + r_n) \quad \text{où les } r_i \text{ sont des pseudo-arborescences.}$$

Soit σ une permutation de $[1 \ldots n]$ qui donne l'ordre de lecture des r_i.

$$\sum_{x_i \mathcal{R} y} d(x_i, y) = d(x_{\sigma(1)}, y) + \ldots + d(x_{\sigma(n)}, y)$$
$$= |r_{\sigma(1)}| + 2|r_{\sigma(2)}| + \ldots + (n-1)|r_{\sigma(n)}|$$

Il est clair que cette quantité est minimale quand la fonction de lecture sélectionne les sous-branches de r dans l'ordre des tailles décroissantes. Cependant, le gain d'une séquence étant prioritaire (contrainte (4)), nous avons perturbé ce choix par la recherche de la sous-branche r_k.



Je l'ai dressé : chaque fois que je sors du labyrinthe, il me donne du fromage.

CHAPITRE 4 : Acquisition d'un contenu didactique

4.1. INTRODUCTION.

Après avoir étudié, au chapitre précédent, le point de vue de l'enseignant ou du gestionnaire de formation (on pourrait dire, à l'instar des québécois, du médiatiseur), nous abordons dans ce chapitre le point de vue de l'enseigné, dans le cadre de l'acquisition *individuelle* de connaissances, de techniques, de phénomènes et de lois , ... qui constituent encore une part importante des processus d'apprentissage pendant une formation initiale ou continuée. Plus précisément, nous nous intéressons aux différentes fonctions que peut remplir l'ordinateur lorsque l'étudiant l'utilise de manière interactive, ce qui recouvre, on l'a vu au chapitre 2, les applications suivantes :

- interrogation de banques de données,
- enseignement assisté par ordinateur de type tutoriel,
- gestion de cheminement,
- exercices répétés.

On a écrit relativement peu sur des théories de transmission de contenus. Les études faites se répartissent entre des théories stochastiques de l'apprentissage [53,176,212] ou des modèles déterministes adaptatifs : ce sont ces derniers que nous avons étudiés. Mais le plus souvent les praticiens (ou les vendeurs de logiciel) proposent un système en en fournissant l'anatomie et non les objectifs pédagogiques sous-jacents, par exemple [46] :

"Il ne faut pas voir dans la CMEAO une option pédagogique précise. Choisir une stratégie est affaire de pédagogue et non d'informaticien. La CMEAO se borne à être une infrastructure souple et simple d'emploi, laissant aux enseignants le soin de l'utiliser comme un auxiliaire de l'enseignement sans se soucier du substratum informatique qui la sous-tend.

Un moden (module d'enseignement) est plus une notion informatique que pédagogique. Du seul point de vue pédagogique, un moden s'apparente à une leçon portant sur un sujet bien déterminé et unique. Par exemple, la réponse à la question : "je voudrais étudier le théorème de pythagore" constitue un moden composé par exemple de l'énoncé du théorème, de la démonstration, d'un exemple et d'exercices.

Mais il est évident qu'une même notion peut être enseignée selon des niveaux d'approfondissement différents selon le bagage intellectuel de l'élève. On n'enseigne pas la loi d'Ohm au lycée de la même façon qu'à l'université.

Par définition, un macromoden est un ensemble de modens traitant tous d'un même sujet, mais selon des niveaux d'approfondissement différents".

La plupart, en fait, ont reçu leur système à partir de l'enseignement programmé, de manière analogue à celle qui consiste à "patcher" un système d'exploitation : on ajoute ceci, puis cela, etc.. En ce qui nous concerne, nous avons adopté une démarche un peu plus scientifique (nous ne voulions pas écrire un système mais apporter une contribution méthodologique) en partant d'une étude de l'enseignement programmé. L'essentiel de cette étude, qui a servi de base à un enseignement de troisième cycle de didactique des mathématiques, est exposé au paragraphe 4.2. Nous avons ensuite élaboré un modèle de système [169], dont une version révisée est présentée au paragraphe 4.3. Le paragraphe 4.4. décrit l'utilisation de ce système dans les quatre types d'application précédemment cités , à savoir : interrogation de banques de données, E.A.O. de type tutoriel classique, gestion de cheminement et exercices répétés. Le chapitre 6 décrit une utilisation dans ce que nous appelons au paragraphe 4.2. le macro-E.A.O.

4.2. MODELE MATHEMATIQUE DE L'ENSEIGNEMENT PROGRAMME ET DE L'ENSEIGNEMENT ASSISTE PAR ORDINATEUR.

Cette étude résulte de la traduction, de l'adaptation et d'une généralisation personnelle de différents articles.

4.2.1. *Structure de l'item et du programme : une première formalisation.*

4.2.1.1. Définitions inruitives [PROCHNOW,63]

L'analyse du contenu d'une matière à enseigner aboutit à un ensemble fini Q dont chaque élément est formé d'une unité d'information et d'une question (l'auteur distingue ces deux parties afin de pouvoir dire que chacune d'elles peut être vide, mais ce n'est pas utile dans la construction ultérieure du modèle). De même, l'expérience professionnelle des enseignants chargés de la rédaction du programme permet d'élaborer un ensemble fini R de réponses "plausibles" aux questions. Rédiger un programme consiste donc d'une part à répartir l'information et les questions dans des unités physiques appelées items, d'autre part à décider du cheminement d'un élève à travers les items du programme en fonction des réponses qu'il donnera aux questions posées. En première approximation, on est donc conduit à considérer un programme comme un ensemble P d'items. Chaque item "contient" un élément de Q, autrement dit il existe une application surjective de P dans Q.

D'autre part, un item admet généralement un item successeur qui dépend de la réponse fournie par l'élève. Autrement dit, pour chaque item p, il existe un certain nombre de réponses possibles. Les couples (p,r) où r est une réponse possible de l'item p constituent une partie D de $P \times R$; on se donne une application s de D dans P qui associe au couple d'un item et d'une réponse à cet item, l'item suivant à lire.

Tout ensemble c de réponses possibles à un item p est la réunion de deux parties disjointes, les réponses "justes" et les réponses "fausses". Parmi les réponses fausses figure toujours une réponse de la forme "une autre réponse" ou "je ne sais pas".

Ceci conduit à la définition formelle suivante :

4.2.1.2. Formalisation.

Un *programme* est la donnée de trois ensembles finis Q, R et P, d'une application surjective q de P dans Q, d'une partie D de $P \times R$ et d'une application s de D dans P.

Les éléments de P sont appelés *items*.

L'ensemble des *réponses* d'un item p, égal à l'ensemble

$$c(p) = \{r \in R \mid (p,r) \in D\}$$

est partitionné en réponses fausses et réponses justes.

4.2.1.3. Application à trois types "classiques" de programmes.

- Programmes linéaires (Skinner).

L'ensemble Q et la fonction q sont choisis de façon que l'élève ne puisse pas se tromper. L'ensemble des réponses fausses à un item est donc vide et pour tout $p \in P$, $c(p) = \{r\}$. D est donc en bijection avec P privé d'un élément et l'application s détermine un ordre total dans P dans lequel $s(p,r)$ est le successeur de p .

- Machines de Pressey.

L'ensemble P est totalement ordonné, comme dans le cas précédent.
L'ensemble D étant choisi, pour tout couple (p,r) appartenant à D on a

$$s(p,r) = p \text{ ou successeur de } p .$$

L'ensemble $c(p)$ des réponses associées à un item est la réunion de deux parties disjointes :

$$V(p) = \{r \in R \mid s(p,r) = \text{successeur de } p\} \quad \text{(réponses justes à } p\text{)}$$

$$F(p) = \{r \in R \mid s(p,r) = p\} \quad \text{(réponses fausses)}$$

Cet ensemble est présenté de façon explicite dans l'item (QCM). L'erreur n'est pas expliquée.

- Programmes ramifiés (Crowder)

L'ensemble P est la réunion de deux parties disjointes P_p (ensemble des items principaux) et P_d (ensemble des items de dérivation). P_p est totalement ordonné. On suppose que R contient la réponse vide notée ε et D est alors partitionné en deux sous-ensembles inclus respectivement dans $P_p \times R$ et $P_d \times \{\varepsilon\}$. Pour tout item principal p tel que $(p,r) \in D$, $s(p,r)$ est égal à successeur de p ou appartient à P_d ; on définit alors

$$V(p) = \{r \in R \mid s(p,r) = \text{successeur de } p\}$$
$$F(p) = \{r \in R \mid s(p,r) \in P_d\}$$

on suppose que pour tout $r \in F(p)$:

$$s(s(p,r), \varepsilon) = p .$$

Dans un item de dérivation p, $q(p)$ est remplacée par un commentaire sur la réponse fournie précédemment par l'élève. Il faut donc modifier la définition de l'ensemble Q.

Dans un certain sens que nous allons préciser, un programme ramifié peut être comparé à une machine de Pressey.

Soit δ l'application de P_p dans l'ensemble $\mathcal{P}(P_d)$ des parties de P_d qui à chaque item principal associe ses irems de dérivation. Pour tout $r \in F(p)$ on a $s(p,r) \in \delta(p)$.

Un *super-item* est un couple $(p,\delta(p))$ avec $p \in P_p$. L'ensemble Π des super-items est une partie de $P_p \times \mathcal{P}(P_d)$. Il existe une application $\hat{q}$ de Π dans $\mathcal{P}(Q)$ définie par

$$\hat{q}(p,\{p_1,p_2,\ldots,p_n\}) = \{q(p),q(p_1),\ldots,q(p_n)\}.$$

L'ensemble Π est totalement ordonné par l'ordre de P_p, donc il existe une fonction successeur de Π (privé du dernier super-item) dans Π. Soit Δ la partie de $\Pi \times R$ définie par

$$((p,\delta(p)),r) \in \Delta \Leftrightarrow (p,r) \in D .$$

Pour tout couple $((p,\delta(p)),r) \in \Delta$, l'application σ est donnée par

$$\sigma((p,\delta(p)),r) = (p,\delta(p)) \text{ ou successeur de } (p,\delta(p))$$

selon que la réponse est juste ou fausse.

On peut ensuite définir $V(p,\delta(p))$ et $F(p,(\delta(p))$ comme dans le cas des machines de PRESSEY.

4.2.1.4. Progrès réalisés grâce à l'utilisation de l'ordinateur.

L'ordinateur permet de supposer l'ensemble des réponses de l'élève "infini", dans le sens que le questionnaire à choix multiples, d'explicite, devient implicite, l'élève étant invité à fournir *une réponse construite*. Le rôle de l'analyseur de réponses est de ramener chaque réponse d'élève à une réponse prévue, c'est-à-dire un élément de R. Toute réponse non reconnue est assimilée à la réponse "je ne sais pas", mais elle est enregistrée et peut conduire à une extension de l'ensemble R prévu a priori. C'est en quelque sorte le point de vue de l'observateur, selon VERMERSCH [201], qui est introduit ici.

4.2.2. *Algorithmes d'enseignement.*

Reprenons la définition de 4.2.1.2. avec les trois ensembles Q, R et P.

Dans P, on distingue l' ensemble ω des items terminaux (un item terminal ne contient pas de question).

Les éléments du monoïde libre sur R, noté R^*, seront appelés *comportements* et notés.

$$r = r_1\, r_2 \ldots r_t$$

Si on pose $Y = P \setminus \omega$, on appelle *cheminement* d'un élève dans le programme P un objet de la forme

$$e = y\omega_j \quad \text{où} \quad y = y_1 y_2 \ldots y_t \in Y^* \qquad \text{et} \qquad \omega_j \in \omega^* .$$

Le cheminement est dit *incomplet* si $\omega_j = \Lambda$, *complet* si $|\omega_j| = 1$ (longueur du mot ω_j), *redondant* si $|\omega_j| > 1$.

L'ensemble des cheminements est noté E.

Par convention, on pose $|e| = t + |\omega_j|$.

Nous présentons deux modèles d'algorithme d'enseignement. Le premier, le plus général, travaille sur les ensembles définis ci-dessus. Le second se restreint à un sous-ensemble de l'ensemble des comportements (comportements dits *sensés*).

Les deux démarches ne sont pas complètement différentes.

4.2.2.1. Algorithme d'enseignement au sens large. [FRANK, 63]

Un algorithme d'enseignement au sens large est un triplet

$$(R,P,\varphi)$$

où φ est une application de R^* dans E qui vérifie les conditions suivantes :

(AE1) $(\forall r \in R^*)$ $|\varphi(r)| = |r|$ (le cheminement comporte autant d'items que le comportement de réactions)

(AE2) $(\forall r \in R^*)$ r' facteur gauche de $r \Rightarrow \varphi(r')$ facteur gauche de $\varphi(r)$

On note $\varphi(r) = \varphi(r')e_{r'}(r)$.

(c'est une condition de déterminisme : si on prolonge un comportement, le cheminement correspondant au prolongement ne diffère pas, au début, du comportement initial).

(AE3) $(\forall y \in Y)$ $(\exists r_i \in R^*)$ $\varphi(r_i)$ se termine par y

(tout item du programme est atteint)

(AE4) $(\forall r \in R^*)$ $(\exists r' \in R^*)$ $\varphi(rr')$ est complet

(il n'existe aucun item dont l'élève ne puisse sortir, ni de possibilité de boucler sans fin dans le programme.

4.2.2.2. Algorithme d'enseignement au sens strict. [LANSKY,63]

Un algorithme d'enseignement au sens strict est un quadruplet

$$(R,P,\varphi^*,\sigma)$$

dans lequel une fonction $\sigma : P \to \mathcal{P}(R)$ définit une fois pour toutes les réactions "sensées" - c'est-à-dire prévues par le programme - à l'issue d'un item : on retrouve l'ensemble D de la première formalisation.

La fonction $\varphi^* : R^* \to E$ n'est plus partout définie. Plus précisément, elle n'est définie que sur des *comportements sensés*, c'est-à-dire tels que

$$r_i \in \sigma(y_i)$$

où y_i est l'élément de rang i de $\varphi^*(r)$, les comportements sensés se définissant par récurrence à partir des comportements sensés de longueur 1. Sur son ensemble de définition, φ^* doit encore vérifier (AE1), (AE2) et (AE3).

Si on appelle X la fonction de l'ensemble des comportements sensés dans P qui à $r = r_1r_2\ldots r_t$ associe l'item y_{t+1} auquel l'élève va être envoyé, la condition (AE4) ne sera vérifiée qu'en imposant à X certaines propriétés, destinées en particulier à éviter un bouclage infini.

Les cheminements obtenus à partir des comportements sensés sont appelés *cheminements sensés*.

L'auteur démontre les résultats suivants :

(1) la construction de φ^* est toujours possible à partir de R, P et σ.

(2) φ^* peut être prolongée en une application φ définie sur R^* (par exemple par $\varphi(r\rho) = \varphi^*(r\rho)$ si φ^* est définie sur $r\rho$, $r \in R^*$, $\rho \in R$

$\varphi(r\rho) = \varphi^*(r)\,\omega_j$ si φ^* est définie sur r

$\varphi(r\rho) = \varphi(r)$ si φ^* n'est pas définie sur r)

(3) Pour tout φ et tout σ, il existe une application φ^* unique se prolongeant en φ sur R^* .

4.2.3. *Algorithmes d'enseignement de MARKOV.* [FRANK, 63]

Deux comportements r_1 et r_2 sont dits *équivalents* si $\varphi(r_1)$ et $\varphi(r_2)$ se terminent par le même item et on note

$$r_1 \approx r_2$$

Un algorithme d'enseignement est dit de MARKOV si

$(\forall r_1, r_2 \in R)$ $r_1 \approx r_2 \Rightarrow (\forall r \in R^*)$ $e_{r_1}(r_2r) = e_{r_2}(r_2r)$

(les notations sont celles de la condition (AE2)).

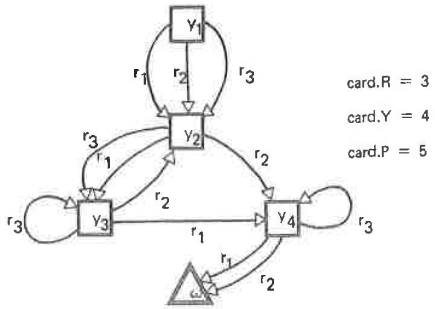
Autrement dit : deux comportements différents ayant engendré des cheminements différents mais se terminant au même item vont, s'ils sont prolongés par le même comportement, engendrer une même suite de cheminements.

On voit qu'il existe alors une fonction $L : P \times R \to P$ telle que

$$y_{t+1} = L(y_t, r_{t+1})$$

c'est-à-dire que l'item qui va suivre est parfaitement déterminé par celui qui l'a précédé et la réaction qu'il a suscitée.

Un algorithme d'enseignement de MARKOV peut être représenté par un graphe dont les sommets représentent les items et les arcs les réactions. Voici un exemple :



4.2.4. *Description des principaux types de programmes* [FRANK, 63]

Soit la relation binaire dans P

$y_i \prec y_j \Leftrightarrow (\forall r \in R^*$ tel que $\varphi(r)$ se termine par y_i, il existe $r' \in R^*$ tel que $\varphi(rr')$ se termine par $y_j)$.

Cette relation est transitive.

On dit qu'un algorithme d'enseignement de MARKOV est *sans circuit* si

$$(\forall y \in P) \quad \neg(y \prec y) \ .$$

Un A.E.M. est dit *linéaire* si

$(\forall y_i, y_j \in P) \quad y_i \neq y_j \Rightarrow y_i \prec y_j$ ou $y_j \prec y_i$.

Un A.E.M. non linéaire est dit *ramifié*.

Un A.E.M. est dit *directif* si

$(\forall y \in P)$ $(\forall r \in R^*$ tel que $\varphi(r)$ se termine par $y)$ $\{e_r(r\rho)\}_{\rho \in R}$ ne contient qu'un seul item différent de y.

Un A.E.M. non directif est dit *adaptatif*.

Compte-tenu de ces définitions, les principaux types d'enseignement programmé peuvent se répartir en six grandes catégories (figure 15).

4.2.5. *Automates d'enseignement pour A.E.M.* [FRANK, 63]

Un automate [4] est un quintuplet

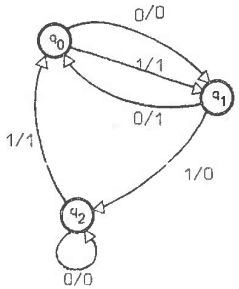
$$M = (X, Z, Q, \delta, \lambda)$$

dans lequel X est l'ensemble fini des *entrées*, Z l'ensemble fini des *sorties*, Q l'ensemble des états ; δ (fonction de transition des états) est une application de $Q \times X$ dans Q ; λ (fonction de transition des sorties) est une application de $Q \times X$ dans Z.

Un automate est donc la description mathématique d'une machine qui étant à l'instant t dans l'état q et recevant x passe à l'instant $(t+1)$ dans l'état $\delta(q,x)$ et émet $\lambda(q,x)$.

Un automate est dit *fini* quand Q est fini.

Un automate fini peut se représenter par un graphe dont les sommets sont les états et les arcs les couples (entrée, sortie) ; par exemple :



$Q = \{q_0, q_1, q_2\}$

$X = \{0,1\}$

$Z = \{0,1\}$

$\delta(q_0, 0) = q_1$

$\lambda(q_0, 0) = 0$

$\delta(q_0, 1) = q_1$ etc...

Un *automate de Moore* est un automate dont la sortie ne dépend que de l'état final, c'est-à-dire qu'il existe une application $\beta : Q \to Z$ telle que :

$$\lambda(q,x) = \beta(\delta(q,x)).$$

Pour un automate de Moore, on peut donc prolonger δ et λ à $Q \times X^*$ de la façon suivante :

$$\delta(q, \Lambda) = q$$

$$\lambda(q, \Lambda) = \beta(\delta(q, \Lambda)) = \beta(q)$$

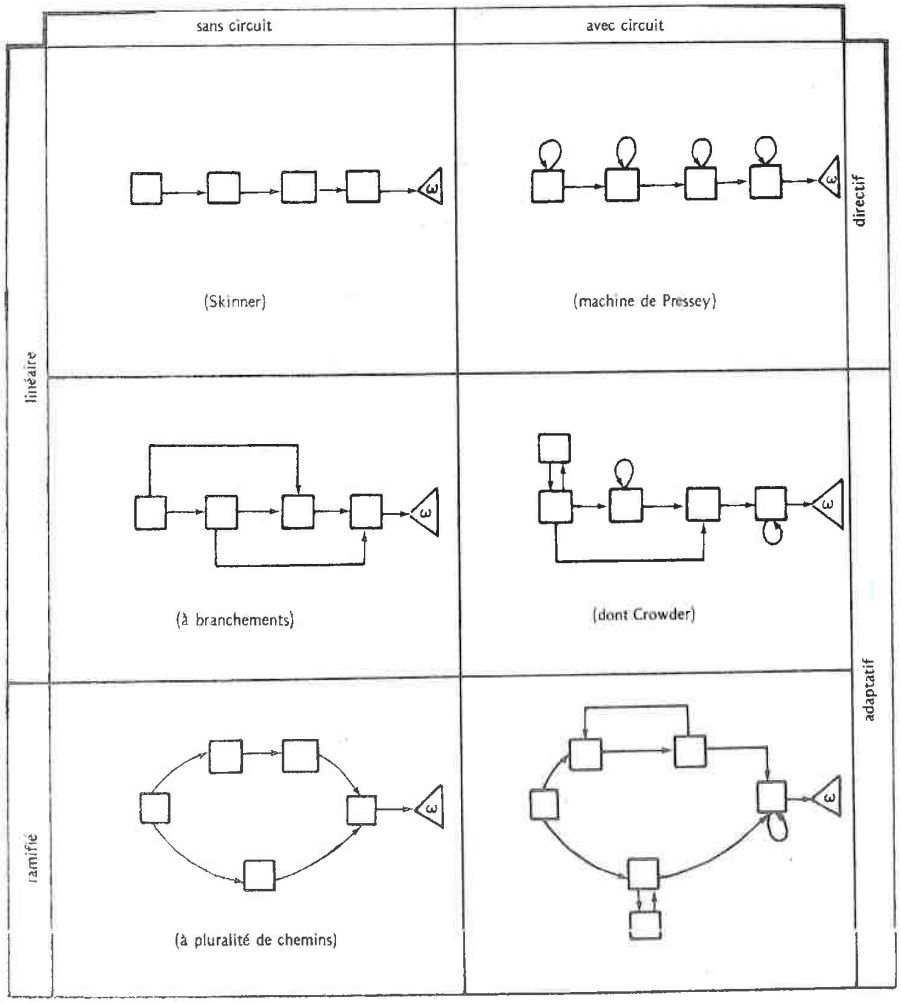


Figure 15 : Classification des types d'enseignement programmé selon trois critères.

$$\delta(q,xx') = \delta(\delta(q,x),x')$$

$$\lambda(q,xx') = \beta[\delta(q,xx')] = \lambda(\delta(q,x),x')$$

On a tout naturellement le résultat suivant :

Tout algorithme d'enseignement de MARKOV est associé à un automate de MOORE avec $Q = Z = P$, $X = R^*$, $\delta = L$ et β = identité.

Souvent, en enseignement programmé, des couples (item, réaction) différents envoient au même item mais cependant c (commentaire sur la réaction) est différent. Le modèle le mieux adapté est donc celui d'un automate quelconque dont les sorties sont composées de deux parties, le commentaire et l'état final. On a alors deux possibilités de transformation du modèle précédent :

* Remplacer l'automate par un automate de MOORE équivalent : il suffit de multiplier les états (autant d'items différents que de réactions différentes) et de faire $Z = Q$.

En effet, étant donné un automate M, on peut construire un nouvel automate

$$\hat{M} = (X,Z,\hat{Q},\hat{\delta},\hat{\lambda})$$

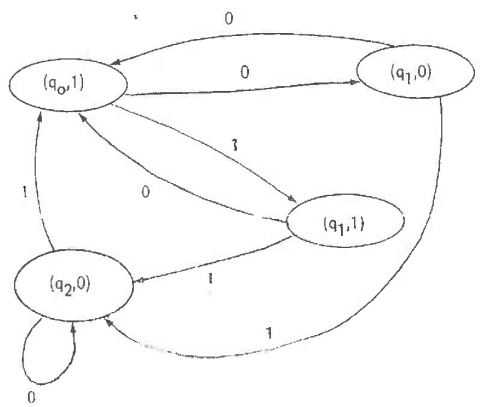
où $\hat{Q}$ est l'ensemble obtenu en séparant chaque $q \in Q$ en plusieurs états $(q,y) \in Q \times Z$, un pour chaque sortie qui peut être associée à une transition vers cet état q :

$$\hat{Q} = \{(q,y) \mid \exists\, q' \in Q \text{ et } x \in X \text{ tels que } \delta(q',x) = q \text{ et } \lambda(q',x) = y\},$$

$$\hat{\delta}((q,y),x) = (\delta(q,x),\lambda(q,x))$$

$$\hat{\lambda}((q,y),x) = \lambda(q,x).$$

La fonction $\hat{\beta} : \hat{Q} \to Z$ qui à (q,y) associe y vérifie bien $\hat{\lambda} = \hat{\beta} \circ \hat{\delta}$, donc $\hat{M}$ est un automate de MOORE. Notre exemple a comme automate de MOORE équivalent :



* Associer deux automates, un quelconque et un de MOORE, qui ont le même ensemble d'états, les sorties du premier étant les commentaires et les sorties du second les états eux-mêmes :

$$\lambda(y_t,r_{t+1})) = \lambda_1(y_t,r_{t+1}),\ y_{t+1})$$

4.2.6. *L'E.A.O. de type tutoriel classique.*

Il fonctionne comme un automate de MOORE, éventuellement obtenu après transformation de l'automate de départ. En effet, la plupart des systèmes se disent "adaptatifs", c'est-à-dire que l'item où est envoyé l'étudiant ne dépend pas uniquement de sa dernière réponse, mais aussi d'un certain nombre de renseignements sur ses réponses ou son cheminement antérieur qui ont été mémorisés (par le biais d'indicateurs ou de compteurs, voir par exemple [145]).

Ce type d'organisation apparaît dans le langage-auteur (cf. paragraphe 2.4.2), souvent de manière explicite [46,57,145,159], pas du tout lorsque la séquence d'E.A.O. est écrite dans un langage universel (mais je sais par expérience que retrouver cette organisation est alors possible [98]), un peu dans les langages intermédiaires entre les langages-auteur et les langages universels [128].

La stratégie d'enseignement conditionne donc complètement l'apprentissage, même si l'adaptativité du système est très élaborée.

4.2.7. *Les extensions possibles. Le macro-E.A.O.*

L'extension la plus significative de l'E.A.O. de type tutoriel est son couplage à un système documentaire (banque de données). L'idée n'est pas nouvelle. On la trouve par exemple dans [138] :

"(Le texte scientifique) en effet peut être considéré comme une intégration de deux niveaux d'énoncés :

- un niveau théorique constitué par une collection d'affirmations générales dont le rapport mutuel minimum est la non-incomptabilité et dont le rapport maximum est la possibilité d'une déduction, lorsque le langage propre utilisé offre les garanties nécessaires.
- un niveau subordonné, qui est constitué par le récit des manipulations non verbales servant de justification expérimentale aux affirmations générales de niveau précédent.

Ces considérations nous amènent à proposer un "système de transmission de connaissances scientifiques" qui serait le suivant :

a) dictionnaire

b) recueil d'affirmations théoriques

c) recueil de débats métascientifiques.

L'utilisation du système... peut se concevoir de diverses façons :

a) comme système d'enseignement
b) comme système documentaire".

Elle est opérationnelle dans [46], qui propose une statégie directive (automate de MOORE simple), une stratégie non directive (l'étudiant interroge la banque de données en langage naturel) et une stratégie mixte.

L'autre extension possible est celle de l'enrichissement des items, dont on a parlé en 2.5. Cet enrichissement présente plusieurs intérêts :

- il permet, on l'a déjà dit, un apprentissage moins mécaniste que l'enseignement programmé.
- il fait prendre tout son sens à la notion de "retour en arrière" : dans ce cas, on ne se rebranche pas sur le même texte d'un petit item, mais éventuellement sur une autre version d'un item quantitativement plus important. Dans [46], on trouve une notion voisine mais différente, celle de stratégie d'approfondissement liée à la notion de macromoden.
- il rapproche l'E.A.O. de la gestion de cheminement, un item pouvant devenir, à la limite, un module non transmis à l'aide de l'ordinateur.
- il donne tout son sens au couplage à un système documentaire : interroger une banque pour récupérer une phrase ou deux ne présente en effet aucun intérêt, c'est toute la différence entre un dictionnaire et une encyclopédie.

Nous appellerons *macro-E.A.O.* un tel E.A.O. dérivé du type tutoriel. Nous y avons pour notre part introduit deux autres fonctions qui nous semblent importantes et que nous n'avons paq rencontrées jusqu'ici dans des systèmes existants : une fonction d'aide à la synthèse et une fonction d'adaptation d'un cours au cas particulier de chaque étudiant.

Si de plus on impose un système de laisser à chaque instant l'étudiant libre de choisir (puis modifier) l'utilisation qu'il veut faire dudit système, il devient alors possible d'appeler un tel système non plus un système de transmission d'un contenu [138] mais un système *d'acquisition d'un contenu*, le dialogue avec l'ordinateur devenant un élément, parmi d'autres, de la situation d'apprentissage dans laquelle se trouve l'étudiant.

4.3. UN SYSTEME D'ACQUISITION DU CONTENU.

Nous décrivons dans ce paragraphe d'une part les objets didactiques manipulés par le système, d'autre part les fonctions du système opérant sur ces objets, que nous appellerons des *modes*.

4.3.1. *Objets*.

Le système manipule :

* des informations au sens donné en 3.3.1. Compte-tenu du fait que nous nous rapprochons de l'informatique (un tel système va déboucher sur des programmes et des données, data en anglais, dont la traduction en information est plus exacte), nous ne pouvons plus les appeler ainsi. A défaut d'un autre mot, nous les avons appelés des *concepts*. Il s'agit d'une notation qui n'a d'intérêt que pour l'usage que nous en faisons ! Un concept peut contenir plusieurs parties, fonction de deux attributs, le contexte et la facilité.

  Le *contexte* est l'indicateur du public. Un concept pourra être présenté avec des exemples et exercices d'illustration différents, un langage adapté à l'âge ...

  A l'intérieur d'un contexte donné, une information peut être présentée avec plus ou moins d'explications. Par exemple, dans une démonstration de théorème, on peut expliciter ou non les déductions de routine, (transitivité de l'égalité...). Ceci fournit un ordre total sur les différentes *versions* à l'intérieur d'un contexte. La première version est la version de *rappel*, qui dans l'exemple ci-dessus ne contient que l'énoncé de la définition ou du théorème.
* des informations destinées à tester l'acquisition des concepts, et que nous appellerons des *questions*, même si elles peuvent être des exercices ou des problèmes. Pour chaque question, on connaît :
  - le concept après lequel elle est susceptible d'être posée.
  - les actions à entreprendre après qu'on l'ait posée : analyseur qu'il faut activer pour analyser la réponse fournie par l'étudiant, ensemble éventuel de commentaires associés aux diverses réponses (l'analyseur ayant pour rôle d'assimiler chaque réponse de l'étudiant à un élément de R appelé réponse prévue - cf. 4.2.1.4. -), les concepts à "pénaliser" à la suite d'une erreur due à leur mauvaise assimilation.
* des enchaînements-type de concepts et de questions associées que nous appellerons des *leçons*.

  De tels enchaînements sont construits par des enseignants, qui peuvent utiliser des concepts ou des questions déjà créés, ou en créer de nouveaux. Ils peuvent être conçus manuellement "après un gros effort de réflexion" [53], ou avec l'aide de l'ordinateur, selon des techniques inspirées de celle qui fait l'objet du chapitre 3.
* des informations relatives aux *étudiants* qui utilisent le système, réparties en
  - éléments d'identification.
  - éléments relatifs à des couples (concept, étudiant), par exemple quelle est la version d'un concept qui a été présentée à l'étudiant en dernier, quelle est la

"connaissance" qu'un étudiant a d'un concept.

- éléments relatifs aux réponses des étudiants aux questions.
- éléments relatifs à des couples (leçon, étudiant), par exemple l'endroit où l'étudiant quittant le terminal a interrompu un quelconque des cours dans lesquels il est inscrit.
- éléments relatifs aux comportements des étudiants dans le système, identifiables s'il s'agit de s'en servir dans une évaluation personnelle des étudiants, anonymes s'il ne s'agit que d'avoir une rétro-action sur le médium ou des éléments destinés à une recherche en sciences de l'éducation (cf. 1.4.)
- interventions des étudiants relatives à leur travail.

La présence ou l'absence d'une partie de ces informations constitue une différence importante entre la gestion de cheminement et l'E.A.O. de type tutoriel. Mais qui peut le plus peut le moins, et un système qui se veut un tant soi peu universel doit contenir ces informations d'une part, et d'autre part prévoir des interruptions (en E.A.O. tutoriel, à de rares exceptions près [26], on vient prendre une leçon d'une heure et c'est fini !)

4.3.2. *Fonctions*.

Le système se comporte comme un automate de MOORE (cf. 4.2.5), dont les entrées sont

- des interventions de l'étudiant relatives à l'utilisation qu'il veut faire du système (passage d'un mode à un autre)
- des réponses de l'étudiant à des questions dont le système a besoin pour son fonctionnement "logique", par exemple, dans la fonction "tutorielle" décrite ci-après, quelle est la leçon sur laquelle l'étudiant veut travailler, si plusieurs leçons sont en cours.
- les réponses de l'étudiant aux questions décrites en 4.3.1.

Les sorties de l'automate sont :

- le texte des concepts (au sens de 4.3.1)
- le texte des questions ou des commentaires associés
- des items de synthèse (voir ci-après)
- des messages et questions destinés au fonctionnement "logique".

Les transitions sont décrites à l'intérieur de chacun des modes ci-après.

4.3.2.1. Prologue

Ce mode contient toute la séquence d'identification de l'étudiant et d'initialisation de certains paramètres.

4.3.2.2. Mode documentaire

L'étudiant chemine librement dans les concepts. Il ne résoud pas de problèmes.

4.3.2.3. Mode adaptation de cours

L'étudiant a choisi de s'inscrire à une leçon. Ce mode lui permet d'adapter la leçon à son profil : choix ou modification de certains paramètres, court-circuit éventuel de certains concepts de la leçon qu'il connaît déjà, et cela à n'importe quel moment de la leçon.

4.3.2.4. Mode tutoriel

Dans le mode tutoriel, et dans un but d'universalité, on considère le contenu d'un concept comme une "boîte noire". A la limite, l'ordinateur peut ne pas présenter les informations, mais seulement poser des questions, les informations étant acquises ailleurs (cours oral, livre, film...). A mi-chemin, l'ordinateur peut présenter lui-même les informations sur des supports variés (écran cathodique, magnétophone, ...). Au mieux, les informations seront elles-mêmes des modules d'enseignement assisté par ordinateur, par exemple des simulations.

Pour chaque concept, la séquence est la suivante :

- "présentation" du concept i
- résolution d'un certain nombre de questions associées à ce concept ; au moment de la traduction par l'analyseur de réponse d'une réponse en réponse prévue, incrémentation éventuelle de compteurs de performances relatifs à l'étudiant pour les concepts mis en relation par la question. Passage éventuel à une révision si la performance sur un concept devient trop mauvaise.

4.3.2.5. Mode récapitulatif

Un item récapitulatif présente les concepts acquis depuis l'entrée dans la leçon, le temps passé, les concepts qui restent à acquérir et le temps approximatif restant à passer.

4.3.2.6. Epilogue

Ce mode termine la session.

Nous ne décrirons pas davantage dans ce chapitre les objets et fonctions du système, car c'est l'objet du chapitre 5 de les préciser dans un contexte de

macro-E.A.O. En effet, son utilisation dans des types d'application différents nécessite, on le verra au paragraphe suivant, des approfondissements qui varient d'une application à l'autre.

4.4. UTILISATION MULTIPLES DU SYSTEME.

4.4.1. *Interrogation de bases de données*.

Il s'agit de l'utilisation unique du mode documentaire. Pour ce mode, il paraît important d'offrir la possibilité aux étudiants de poser des questions en langage naturel au lieu de les obliger à coder leurs demandes d'information ; par exemple la question :

"Puis-je savoir quelque chose sur les suites de Cauchy ?"

devrait donner l'une des deux réponses suivantes :

a) "Il n'y a actuellement rien dans la base sur les suites de Cauchy ; nous enregistrons votre demande"

b) "La base contient plusieurs éléments traitant des suites de Cauchy ; ce sont

(1) .....

(2) .....

(3) .....

Tapez le numéro correspondant à votre demande. Si aucun de ces éléments ne vous convient, tapez le numéro 4."

(L'étudiant tape 4)

"Ecrivez de la façon la plus précise possible votre demande concernant les suites de Cauchy".

4.4.2. *E.A.O. de type tutoriel classique*.

Un concept est un item au sens classique du terme. Une seule question lui est associée. L'analyseur permet d'envoyer un commentaire sélectionné, de mettre à jour comptaurs ou indicateurs et l'examen de l'état de l'automate permet de sélectionner l'item suivant.

Une leçon est la description des transitions de l'automate.

Les réponses des étudiants sont enregistrées à des fins d'amélioration des items et de la leçon.

4.4.3. *Gestion de cheminement*.

Un concept est une indication à l'étudiant du travail qu'il doit fournir.

Les questions évaluent ce travail et suggèrent éventuellement des révisions.

Le mode récaputilatif permet l'élaboration d'un plan de travail, car des concepts indépendants peuvent être suivis en parallèle. Le mode adaptation prend en compte ses acquis antérieurs.

Il faut ajouter au système des procédures de gestion des ressources : nous n'avons pas approfondi cette question.

4.4.4. *Exercices répétés*.

Les concepts n'ont pas de contenu : il n'y a que des questions. Il est bon quelquefois de prévoir une procédure "au secours" que l'étudiant puisse appeler en cas de difficulté, dans ce cas le concept contiendra un texte adéquat.

On peut utiliser des générateurs de questions.

Le nombre de questions qu'on pose peut être évalué de façon dynamique grâce aux performances de l'étudiant aux questions relatives aux différents concepts.

Ce paragraphe montre bien qu'un système d'acquisition du contenu complètement spécifié avant sa réalisation matérielle peut déboucher sur un ensemble d'applications qu'on s'accorde maintenant à reconnaître comme différents, uniquement parce que l'option prise jusqu'à présent pour les implanter ne permet pas de passer facilement de l'application unique choisie comme cible aux autres applications.

"Yes, Master?"

CHAPITRE 5 : Le projet SATIRE : un logiciel de macro-enseignement assisté par ordinateur utilisant une base de données.

Le plan que nous avons choisi pour ce chapitre est inhabituel : il ne commence pas par la description du logiciel SATIRE. Ce dernier n'étant en effet que la particularisation au macro-EAO défini en 4.2.7. des idées avancées dans les chapitres 3 et 4, nous avons préféré, pour que sa description soit mieux motivée, la faire précéder de l'historique du projet et des diverses considérations qui nous ont fait choisir une base de données pour la gestion des informations manipulées. Cependant, le lecteur gêné par cette démarche peut commencer la lecture au paragraphe 5.3.

5.1. HISTORIQUE DU PROJET SATIRE.

Nous avons déjà parlé dans notre introduction de la réticence que nous avions à réaliser concrètement un logiciel d'E.A.O. Cependant, en 1976, après avoir recueilli l'avis des personnes d'horizons variés, nous décidions de demander au SESORI des crédits pour une telle réalisation. Le projet comportait deux phases : la première était la réalisation d'un logiciel minimal permettant de satisfaire rapidement des utilisateurs potentiels, et qui à cette époque aurait convenu à des applications d'E.A.O. tutoriel classique (voir § 4.4.2). La seconde phase prévoyait des recherches sur des domaines encore ouverts, à savoir la communication homme-machine (analyse de la réponse, langues naturelles) et les applications type intelligence artificielle dans le domaine des mathématiques.

Il se trouve qu'à l'époque on ne vivait pas encore en France à l'heure de la télématique [144] : le ministère de l'industrie n'ayant plus vocation à s'occuper d'enseignement (même assisté) a renvoyé la balle au ministère des universités. L'intérêt de ce dernier pour une rénovation véritable de la pédagogie de l'enseignement supérieur n'est pas nouveau, et de nombreuses équipes, particulièrement dans le domaine "technologie de l'enseignement-informatique", ont été soutenues financièrement depuis 1965. Mais depuis quelques années les investissements lourds [54,145] ont cédé la place à une relative dispersion des moyens qui, par ailleurs, restaient globalement constants. C'est ainsi que fin 77 une somme assez modeste nous était allouée, qui couvrait à peine la première phase du projet.

Révisant nos intentions premières, nous avons décidé de réaliser un logiciel :

- qui n'ait pas 10 ans de retard : c'est pourquoi nous avons fait le choix de ce que nous avons appelé le macro-enseignemant assisté par ordinateur.
- qui n'ait pas 10 ans d'avance, puisque nous n'en avions pas les moyens, mais qui soit profitable à un public d'étudiants : c'est ce que souhaite le ministère des universités.
- qui permette de montrer la faisabilité des idées que nous avions exprimées sur l'aide de l'informatique à l'enseignement.

Le projet SATIRE (Système d'Aministration et de Transmissions d'Informations Relevant de l'Enseignement) avait vu le jour.

Dans ce chapitre, nous verrons d'abord les raisons qui nous ont poussés à utiliser, pour la mémorisation des données du projet, une base de données. Nous donnerons ensuite la description des différents aspects du projet SATIRE : construction de la banque de données pédagogiques, construction de cours, dialogue de l'étudiant avec l'ordinateur, administration de la base. La réalisation pratique sera exposée dans les chapitres 7 et 8.

5.2. POURQUOI UNE BASE DE DONNEES ?

Les systèmes d'E.A.O. de type tutoriel classique que nous avons pu analyser en détail, à savoir ceux qui ne sont pas commercialisés, opèrent presque tous sur des fichiers classiques.

- LA+LDA à l'O.P.E. de PARIS VII [145], présente la caractéristique de séparer le cours de la structure du cours (comparable à l'ordre des informations dans SATIRE). On peut donc modifier le cours, ou la structure, ou les deux. A la création du cours, les informations entrées par cartes ou au terminal sont compilées, et deviennent des fichiers, des tables, des modules. Les modifications se font directement sur les fichiers, grâce à un module spéciel du système.

- MAGISTER [128] est un "langage-auteur" interprété. Le fichier-source, qui est un programme dans lequel sont intimement liés les informations à transmettre et le cheminement entre ces informations, est rangé sur disque à partir de cartes ou d'un terminal. Une précompilation le transforme en fichier-objet, et il n'est interprété qu'à l'exécution. Par conséquent, les modifications se font sur le source grâce à un éditeur de textes, et seule la précompilation, peu coûteuse, est à refaire.

- Dans le projet JEAN BERNARD, devenu depuis EAOS [54], il avait été explicitement prévu un ramasse-miettes ([96], LMF, A. ALDEGUER et C. REICHSRATH).

- Les concepteurs du système DIGE [52], qui confient actuellement à l'éditeur de textes de SIRIS 8 le soin de gérer une banque de problèmes sophistiqués de mathématiques, pensent utiliser une base de données si le nombre de problèmes devient trop grand.

- Des recherches originales de gestion d'informations en mémoire ont été faites, mais elles semblent très spécialisées ([96], MAOD, B. MONT-REYNAUD).

Si on se limite à ces exemples, on voit qu'à condition de faire les modifications sur les fichiers quasi-définitifs et de prévoir un système astucieux de ramasse-miettes, ou de travailler directement sur le source, l'utilisation de fichiers classiques n'empêche pas une mise à jour correcte des programmes. Par contre, et on l'effleure déjà dans le projet DIGE, vouloir que l'étudiant accède librement aux informations conduit à se poser la question d'organiser ces informations autrement. La C.M.E.A.O. [46] est bâtie sur ce modèle : nous ne savons malheureusement rien sur son système de gestion des "modens" et "macromodens".

Une première tentative d'utilisation d'une base de données est celle du langage EPMINE opérant sur la base SOMINE [58]. Bien que mise au service d'un modèle d'enseignement assez pauvre, elle affirme déjà l'idée que toutes les informations de l'environnement E.A.O. peuvent être contenues dans la base : concepts, relations, exercices, interventions de l'étudiant, ...

L'application la plus élaborée utilisant une base de données, dans le domaine de l'enseignement, est la MODULATHEQUE de X. CASTELLANI [34]. Cependant, c'est plus une application de gestion qu'une application d'enseignement assisté, en particulier elle est utilisée en traitement par lot.

Après ce tour d'horizon, résumons les raisons pour lesquelles nous avons fait le choix de faire opérer SATIRE sur une base de données :

- il y a incontestablement une question de mode et de disponibilité d'outils (ceci d'ailleurs explique en partie pourquoi les application d'avant 1975 ne l'ont pas fait).

- les différents objets manipulés par le système sont indépendants, il n'y a pas redondance de l'information, et cependant tout s'inscrit dans une structure logique cohérente. L'indépendance, quant à elle, permettant, comme on l'a vu en 4.3,
  - d'utiliser le système pour autre chose que le tutoriel
  - de rendre les informations didactiques de la base indépendantes des cours qui les utilisent : des cours différents peuvent utiliser des concepts communs et des questions communes. Ainsi la base intègre-t-elle les connaissances au fur et à mesure de leur introduction au lieu d'en faire des unités séparées sans intercommunication.

- on a toute facilité pour modifier les informations didactiques, pour exploiter les informations relatives aux étudiants.

- si on regarde les langages-auteur, leur plus grosse partie est un langage de manipulation de fichiers [53] ; ici on s'évite la conception et l'écriture de ce langage, puisqu'il s'agira du langage de requêtes de la base.

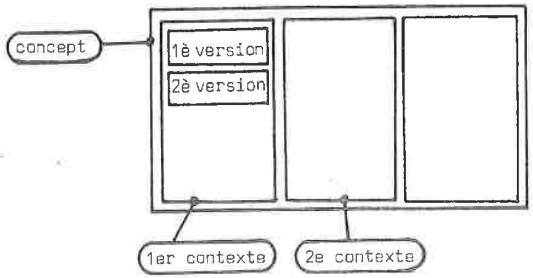
- on fait un pas vers la portabilité du système : changer d'ordinateur revient à changer la description de la structure de la base et les sous-programmes écrits en langage de requête. L'idéal serait, on verra pourquoi un peu plus loin, de décrire les objets de SATIRE en termes de types abstraits [51], mais là se pose un problème de disponibilité de logiciel de base.

5.3. LES OBJETS DU PROJET SATIRE.

Nous reprenons ici les définitions données en 4.3. en détaillant les caractéristiques des différents objets et des relations entre les objets.

5.3.1. *Les concepts*

Un *concept* est une quantité indivisible d'information pédagogique accessible librement par l'étudiant ou pouvant être intégrée à une leçon. Un concept est repéré par un *nom* et appartient à une *discipline*. Tout concept peut posséder plusieurs *contextes*, chacun repérant un *public* donné. Pour chacun des contextes, il peut exister plusieurs *versions* d'un concept donné, dont une version particulière appelée *version de rappel*. Le schéma suivant donne une idée de l'organisation d'un concept :



Chaque version d'un contexte de concept a un contenu, qui est le message qui sera délivré à l'étudiant, et qui peut être un conseil (renvoi à un document écrit ou tout autre medium), une simulation, un texte significatif, ...

A chaque concept est attaché l'ensemble des autres concepts de la base qu'il faut connaître pour aborder son étude, c'est-à-dire de ses antécédents au sens de la relation $\mathcal{R}$ définie au § 3.3.1. Si cet ensemble est vide, le concept est appelé *prérequis* de la base, il contient une unique version dans un seul contexte, qui est un message particulier.

5.3.2. *Les leçons*

Une leçon est repérée par son *nom* et un texte plus important appelé *titre*. Elle est valable pour un *contexte* donné. On appelle *fonction d'ordre* l'enchainement des concepts de la leçon. Un même concept peut figurer plusieurs fois dans la même leçon (cf. § 3.4.). Un concept peut appartenir à plusieurs leçons.

Un concept qui n'est pas un prérequis de la base peut être considéré comme *prérequis d'une leçon* : cela exprime le niveau de départ des étudiants auxquels la leçon est en principe destinée.

5.3.3. *Les questions*

Une *question* possède un *énoncé* et une *solution*. Les questions ne sont pas des objets auxquels on peut accéder directement. Elles sont destinées à tester l'acquisition des concepts, dans un contexte donné et à l'intérieur d'une leçon donnée. On connaît donc, pour chaque contexte de concept d'une leçon, l'ensemble des questions associées. Il faut pouvoir mettre un ordre total sur cet ensemble, si on désire que les questions soient posées dans un certain ordre. Bien entendu, une question peut appartenir à plusieurs leçons.

Des extensions sont à prévoir ici dans le cas où on voudrait se préoccuper de l'analyse automatique de la réponse de l'étudiant. Ce problème ne se posant pas dans le prototype de cours réalisé (cf. chapitre 6), la solution retenue consiste à connaître, pour chaque question, la liste des concepts dont elle met la compréhension en jeu.

5.3.4. *Les étudiants*

Les *étudiants* ont un *nom*, un *prénom* et une *adresse*, qui permettent de les identifier sans ambiguïté. Ils relèvent, en général, d'un des contextes prévus dans le système. Les étudiants peuvent suivre plusieurs leçons en parallèle, on les distinguera en les appelant des *apprentissages* pour lesquels les étudiants pourront choisir des contextes différents.

Pour chaque apprentissage, un certain nombre d'informations relatives à l'endroit où en est l'étudiant sont nécessaires :

- le concept en cours dans la progression normale
- les concepts éventuellement en cours de révision, grâce à une pile
- la version des contextes de concept qu'on lui présente habituellement
- la question en cours
- le nombre de questions qu'on lui pose habituellement après un concept.

En effet, une leçon nécessite en général plusieurs sessions au terminal, et ces renseignements permettent de reprendre une leçon à l'endroit où elle a été interrompue.

Pour chaque version de contexte de concept, on connait l'ensemble des étudiants qui l'ont reçue, à des fins de bonne gestion de la base.

Pour chaque concept on dispose de compteurs de performance de chaque étudiant, afin de pouvoir conseiller des révisions en cas d'erreurs répétées lors des réponses aux questions. Si un étudiant révise un concept, c'est une version différente de celles qu'il a déjà eues qui lui sera présentée, dans le but d'une meilleure compréhension.

Si les étudiants demandent à apprendre des prérequis de la base, cette information est mémorisée, afin de compléter ultérieurement les ressources pédagogiques.

Pour chaque leçon, on mémorisera l'ensemble des étudiants qui l'ont suivie.

Il faut également savoir si une leçon est ou non en cours d'étude par un ou plusieurs étudiants : le problème se posera si on veut la supprimer.

Pour chaque question enfin, on mémorisera des informations sur les réponses des étudiants.

5.3.5. *Les items*

Dans un but d'uniformisation, tous les messages importants envoyés par le système aux étudiants sont regroupés sous ce nom. Un item peut donc être : un message d'explications sur le fonctionnement du système (dans ce cas le mettre hors du programme permet de le modifier plus facilement), un contenu de version de contexte de concept, un énoncé ou une solution de question.

5.3.6. *Les comportements*

La base mémorise, sous forme abrégée, la façon dont les étudiants utilisent le système, ainsi que leur cheminement entre les différents concepts, pour une session donnée.

5.3.7. *Les interventions*

Une originalité de [58] consiste à mémoriser dans la base certaines interventions des étudiants. Nous avons conservé cette idée en distinguant les interventions relatives aux contextes de concepts, aux questions, aux leçons, et au système en général. Les interventions seront régulièrement relevées à des fins d'amélioration des informations du système ou de ce dernier.

5.4. LES FONCTIONS DU PROJET SATIRE.

5.4.1. *Fonctions relatives à l'enseignant ou au chercheur en didactique*

Nous en distinguerons quatre, même si elles seront le plus souvent utilisées en même temps :

- l'introduction dans la base d'un concept, d'une question, d'une leçon.
- la consultation d'un concept, d'une question, d'une leçon de la base.
- la modification d'un concept, d'une question, d'une leçon de la base.
- la suppression d'un concept, d'une question, d'une leçon.
- la *conception assistée* d'une leçon, à partir des objets déjà entrés dans la base (concepts), ou avant leur introduction (questions), selon les algorithmes décrits au chapitre 3.
- la récupération des comportements et des interventions des étudiants.

5.4.2. *Fonctions relatives à l'étudiant*

Il nous suffit de reprendre ici le paragraphe 4.3.2. en le détaillant.

Chaque dialogue de l'étudiant avec le terminal est pilotée par un programme découpé en 6 fonctions appelées modes, suivant le schéma de la figure 16. A l'inverse des autres modes, qui ne comportent qu'une étape, le mode tutoriel et le mode documentaire sont des répétitions d'étapes identiques ; l'étudiant peut donc à la fin de chaque étape, passer d'un mode à l'autre : aucune transition n'est interdite, même si certaines ont une très faible probabilité, car l'étudiant ne peut pas parfaitement connaître en une heure les possibilités du système, et les erreurs sont possibles.

5.4.2.1. Le prologue

Le prologue se déroule de la manière suivante :

- message de bienvenue
- question du système pour savoir si l'étudiant vient pour la première fois

  R = oui création d'un élément de l'ensemble des étudiants et initialisation de certaines relations

  R = non identification de l'étudiant guidée par le système, en prenant note que l'étudiant peut avoir été retiré de la base sans qu'il le sache

- question du système pour savoir si l'étudiant veut connaître les modalités du dialogue, et réponse appropriée dudit système.
- création d'un élément de l'ensemble des comportements.

- attente d'une commande de l'étudiant (pour passage à un autre mode).

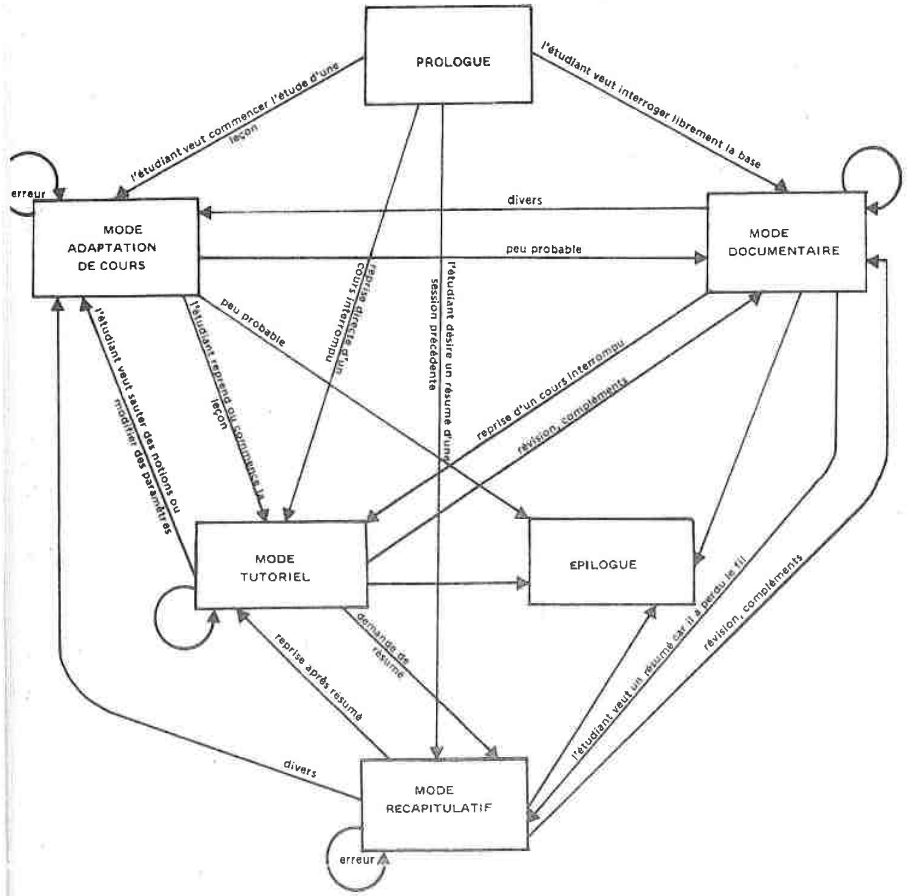


figure 16 : Les modes du logiciel de dialogue de SATIRE

5.4.2.2. Le mode documentaire

Il est accessible de façon naturelle à partir du prologue ou de n'importe quel mode, sauf le mode adaptation de cours. Dans ce dernier cas, on envoie à l'étudiant un message lui signalant qu'il a peut-être commis une erreur, on lui propose le résumé des modalités de dialogue, et le système se met dans l'attente d'une autre commande secondaire.

Pour les raisons que nous avons exposées dans l'introduction de ce chapitre, l'interrogation de la banque de données ne se fait pas, comme dans [46], en langue pseudo-naturelle. L'étudiant dispose donc à côté du terminal d'une liste des concepts présents dans la base (nom et numéro).

Tant que l'étudiant ne demande pas à quitter le mode documentaire, la séquence suivante se répète :

- question du système demandant quel est le concept souhaité ; R = "C1"

C_1 est un prérequis de la base	envoi d'un message à l'étudiant disant qu'on ne peut rien pour lui et mémorisation de la demande
C1 n'est pas un prérequis de la base	. choix du contexte, qui est celui de l'étudiant s'il existe pour l'étudiant et le concept, sinon l'étudiant le choisit parmi ceux qui existent . choix d'une version : si l'étudiant a déjà "vu" ce concept dans une des versions, on choisit la suivante, sinon on choisit la première . envoi au terminal du contenu de l'item qui est le sujet de la version du contexte du concept C1 . mise à jour des relations concept ↔ étudiant . question du système demandant si l'étudiant veut avoir connaissance des leçons dont le concept fait partie (sans en être un prérequis), et réponse appropriée dudit système

- test de sortie du mode.

5.4.2.3. Le mode tutoriel

Le début du dialogue est fonction du mode précédent et du nombre d'apprentissages en cours, selon la table de décision suivante :

	mode précédent	prologue ou m. tutoriel ou m. documentaire			mode récapitulatif		mode adaptation	
conditions	l'étudiant n'a qu'un apprentissage en cours	N	Ø	N	Ø	N	Ø	N
	l'étudiant a plusieurs apprentissages en cours	N	N	Ø	N	Ø	N	Ø
	l'étudiant n'a pas d'apprentissage en cours	Ø	N	N	N	N	N	N
actions	message indiquant une erreur possible et conseillant le mode adaptation	X						
	poursuite de cet apprentissage		X		X		X	celui du mode adaptation
	question pour faire préciser de quel apprentissage il s'agit			X		X		

Tant que l'étudiant ne demande pas à quitter le mode tutoriel et tant que le concept en cours n'est pas le dernier concept de la leçon, la séquence suivante se répète :

- C_1 = concept à apprendre ou réviser (voir ci-après). Le contexte est celui de l'apprentissage.

C1 est un prérequis de la leçon — message l'expliquant et indication éventuelle des leçons auxquelles il appartient et dont il n'est pas un prérequis. Si on "révise" C1, mémorisation de C1 dans l'ensemble des demandes non satisfaites.

C1 n'est pas un prérequis de la leçon

ce n'est pas la 1ère occurrence de C1 dans la leçon et on "apprend" C1 — envoi au terminal de la version de rappel

C'est la première occurrence de C1 dans la leçon ou on "révise" C1

- choix de version (voir mode documentaire)
- envoi au terminal de cette version
- enregistrement d'une éventuelle intervention de l'étudiant

} seulement dans le cas où l'étude de C1 n'a pas été interrompue (voir ci-après)

- procédure des questions

- test de sortie du mode.

C'est la procédure des questions qui permet de comprendre la notion de "révision" d'un concept, c'est donc elle que nous décrirons en premier.

Supposons que l'étudiant aborde pour la première fois le concept C. Si N1 est le nombre de questions auxquelles il a choisi de répondre habituellement, la séquence suivante sera répétée pour i = 1 jusqu'à N1 tant que l'étudiant ne l'interrompra pas et tant qu'il ne commettra pas une erreur nécessitant une révision (A)

- énoncé de la question q_i
- résolution
- analyse conduisant à l'incrémentation éventuelle du nombre d'erreurs commises sur un ou plusieurs des concepts mis en relation dans q_i ; si pour un ou plusieurs des concepts ce nombre d'erreurs dépasse un seuil fixé, ces concepts ainsi que C sont mémorisés dans l'ensemble des concepts à réviser et la présentation des questions est interrompue (A)
- enregistrement d'une éventuelle intervention de l'étudiant

Si après un certain nombre de révisions l'étudiant reprend l'étude de C, les questions i à N1 seulement lui seront posées.

La détermination de C1 se fait de la façon suivante : tant qu'il y a un concept à réviser, C1 est un concept à réviser, sinon c'est le concept suivant le concept en cours dans la leçon.

Bien entendu, la "révision" est un processus récursif. Elle n'est jamais obligatoire, mais le nombre d'erreurs commises sur un concept n'est remis à zéro que si la révision est effectivement faite.

S'il n'y a plus de révision et si le concept en cours est le dernier de la leçon, il y a suppression de l'apprentissage correspondant, envoi d'un message approprié, enregistrement d'une éventuelle intervention de l'étudiant et attente d'une nouvelle commande (pour passage à un autre mode).

5.4.2.4. Le mode récapitulatif

Là encore, le dialogue est fonction du mode précédent et du nombre d'apprentissages en cours, selon la table de décision suivante :

	mode précédent	prologue ou mode documentaire			mode récapitulatif		mode tutoriel ou mode adaptation	
conditions	l'étudiant n'a qu'un apprentissage en cours	N	Ø	N	Ø	N	Ø	N
	l'étudiant a plusieurs apprentissages en cours	N	N	Ø	N	Ø	N	Ø
	l'étudiant n'a pas d'apprentissage en cours	Ø	N	N	N	N	N	N
actions	message indiquant une erreur probable, pas d'exécution	X			X			
	exécution pour cet apprentissage		X				X	
	question pour faire préciser de quel apprentissage il s'agit et exécution			X		X		X

L'exécution consiste en la présentation séquentielle des versions de rappel des concepts acquis jusque là. Si on mémorise d'autres informations dans la base telles que le nombre de concepts restant à acquérir, le temps passé, le temps restant à passer, etc .... ces renseignements peuvent enrichir le mode récapitulatif.

5.4.2.5. Le mode adaptation de cours

Ce mode sert à initialiser ou à modifier les paramètres d'un apprentissage. Il sert aussi à court-circuiter des morceaux de leçons que l'étudiant aurait déjà acquis par ailleurs.
Si l'étudiant vient du mode tutoriel, récapitulatif ou adaptation, la leçon à adapter est en principe la leçon en cours. Le système s'en assure en posant la question, sinon l'étudiant indique de laquelle il s'agit (ce peut être une leçon parallèle ou une nouvelle leçon).
Dans le cas d'une nouvelle leçon, le début de la séquence consiste en la création de l'apprentissage (avec choix du contexte, de la version et du nombre de questions habituellement posées). Dans le cas d'une ancienne, le système interroge l'étudiant pour savoir quels paramètres il veut modifier.
La seconde partie de la séquence est destinée à permettre un raccourci à l'étudiant, en début de leçon (nouvelle leçon) ou en cours de leçon (ancienne leçon).

L'ordre de présentation des concepts est celui du mode tutoriel. On donne à l'étudiant le nom du concept, on lui propose la version de rappel, on lui propose la 1ère question avec sa solution, et on itère avec le concept suivant tant que l'étudiant ne demande pas à changer de mode.

5.4.2.6. L'épilogue

Il précède directement la fin de la session. On mémorise une intervention éventuelle de l'étudiant sur les modalités du dialogue et un message approprié est envoyé.

5.4.3. *Administration de la base de données*

On a déjà constaté que l'enseignant et l'étudiant ont besoin d'un certain nombre de renseignements sur le contenu de la base : liste des concepts, liste des leçons ...

En outre, on ne peut laisser complètement la base en libre-service : les ensembles de concepts, de leçons, de questions, d'étudiants ne peuvent grandir indéfiniment. Il faut donc ôter de la base les étudiants qui sont définitivement partis, supprimer les leçons qui ne servent plus (ainsi que les questions), surveiller l'ensemble des concepts, relever régulièrement les interventions et les comportements.

Il est donc nécessaire de prévoir un administrateur de la base chargé de veiller à sa bonne utilisation. Nous donnons ci-après une liste non exhaustive des "utilitaires" qu'il aura à mettre en oeuvre :

- liste de tous les concepts
- liste de tous les concepts d'une discipline
- liste des étudiants ayant étudié une version de un contexte de un concept, ou un concept
- liste des leçons
- liste des leçons dont fait partie un concept
- liste des leçons auxquelles une question appartient
- liste des étudiants
- ....

"A pencil? What's a pencil?"

CHAPITRE 6 : Didactique de la programmation et utilisation de SATIRE dans son enseignement.

6.1. INTRODUCTION A LA DIDACTIQUE DE LA PROGRAMMATION

Lors des débuts de l'enseignement assisté par ordinateur, de nombreux informaticiens ont été tentés d'utiliser cet outil dans l'enseignement de la programmation. Il s'agissait d'écrire un cours selon les principes de l'enseignement programmé (en utilisant un langage-auteur ou un langage évolué) pour présenter les instructions du langage étudié, et d'assurer un interface avec l'ordinateur utilisé cette fois-ci en tant que calculateur (cf. 2.2.1) pour que l'étudiant puisse écrire de petits programmes et les tester. Quelquefois, il était nécessaire de bricoler un peu le compilateur pour que les erreurs de l'étudiant donnent lieu à des messages plus pédagogiques que les messages habituels. C'était à l'époque un travail technique assez original, de bons exemples en sont fournis par [35,125].

Mais contrairement à ce qui se passait dans d'autres disciplines, où la didactique commençait à être approfondie, et où l'outil enseignement programmé apparaissait souvent trop rudimentaire au regard des phénomènes d'apprentissage, on assista alors au phénomène contraire : l'outil était en avance, car on ne savait pas ce qu'était l'enseignement (ni l'apprentissage) de la programmation.

Comme le rapporte [71], la didactique s'appuie sur la connaissance approfondie, d'une part, de la matière enseignée et de sa structure, d'autre part, du développement psychologique de l'élève et des processus d'appropriation des connaissances. La première est indispensable pour déterminer les concepts, les méthodes et les savoir-faire les plus importants, tandis que le second est indispensable pour déterminer ce qu'il est possible d'enseigner, dans quel ordre, et avec quelles méthodes.

Dix années nous séparent des premières expériences. Au cours de ces dix années, il n'y en a eu, à ma connaissance, pas de fondamentalement différentes ([26] est un enseignement tutoriel de FORTRAN très classique). Mais d'une part la connaissance de la matière, à savoir la théorie de la programmation, a fait d'énormes progrès [122]. D'autre part, la formation d'un nombre croissant de programmeurs dans les universités a fourni un matériau important pour qu'on commence à jeter les bases d'une didactique de la programmation [78,109,182].

Différentes écoles ont vu le jour. Il n'est pas dans mon propos de les décrire ni de les comparer. L'hypothèse de départ, rapidement abandonnée, qui était qu'apprendre à programmer c'était apprendre les instructions d'un langage de programmation, a continué, mis à part le courant de la programmation structurée [37], à influencer les recherches : LCP (logique de construction de programme, [205]), tournée vers COBOL et les programmes de gestion ; tout l'environnement de la naissance de PASCAL [21,210,211] ; l'utilisation croissante de LOGO [80] ; sans

compter les sous-produits de la définition d'ALGOL 68 [66], et toute la floraison des langages dits de haut niveau (et même de très haut niveau) [132].

Autour de Claude PAIR a commencé à se développer à Nancy une école "orientée-problèmes" et non "orientée-langages" : la méthode déductive de programmation [60, 146], qui a donné lieu à de nombreux développements théoriques et didactiques. J'ai donc tout naturellement été conduite à penser qu'on pouvait envisager, maintenant, de reprendre des recherches sur l'enseignement assisté par ordinateur de la programmation.

Comme le fait remarquer HOC [78], l'apprentissage d'une méthode suscite de nouvelles questions : il s'agit de savoir comment former chez le sujet, débutant dans un domaine, les connaissances qui vont lui permettre de la mettre en oeuvre. La méthode elle-même ne peut pas fournir un tel guidage pédagogique, tout juste peut-elle donner une référence au sujet et au pédagogue pour évaluer l'écart de la conduite au modèle à suivre. Dans une perspective pédagogique, il convient donc de définir non seulement la méthode à laquelle la conduite devra se conformer à une étape finale, mais aussi des méthodes intermédiaires qui constitueront des étapes intermédiaires dans l'atteinte de cet objectif. Dans le travail d'équipe qui a été réalisé autour de la méthode déductive, cela a sans doute été ma contribution la plus importante : je ne voudrais cependant pas, en le publiant dans ce travail, m'en attribuer tout le bénéfice.

Ces recherches devaient ensuite se concrétiser par la réalisation de plusieurs cours pour le projet SATIRE. Les deux aspects les plus importants de ces cours, et qui vont être développés dans la suite sont *l'apprentissage en spirale* et *l'apprentissage par la résolution de problèmes*.

6.2. LES CONCEPTS ET LES DIFFERENTES PROGRESSIONS POSSIBLES.

6.2.1. *Les concepts*

Toute méthode de programmation suppose la définition d'un certains nombre de concepts dits *algorithmiques* permettant l'expression de tout problème en un langage intermédiaire entre son énoncé et le programme correspondant, suivant le schéma suivant :

problème → énoncé → algorithme → programme.

En effet, seul ce langage intermédiaire permet de rendre la résolution du problème indépendante du langage dans lequel il sera programmé. Une des originalités de la méthode de programmation déductive est de séparer la phase

énoncé → algorithme

en deux sous-phases grâce à l'utilisation d'un langage d'algorithmes de type déclaratif (ou statique) permettant de débarasser l'analyse de la préoccupation de l'ordre d'exécution, et donc de la mener en partant de ce qu'on connait le mieux dans un problème, à savoir les résultats attendus. Le schéma devient alors :

problème → énoncé → énoncé explicite → algorithme → programme.

Les deux phases qui restent après obtention de l'énoncé explicite doivent être, à terme, entièrement automatisables.

Dans ce qui suit, nous appelons algorithmique "l'art" de fabriquer des algorithmes, et l'adjectif algorithmique caractérisera ce qui a trait à cet art (contrairement à l'habitude qui nomme langages algorithmiques les langages de programmation).

6.2.1.1. Les concepts algorithmiques

Ils sont répartis en deux groupes. Les premiers, liés aux structures de contrôle du programme cible, sont maintenant bien définis. Les seconds, liés aux structures de données du programme, sont encore actuellement dans une phase de recherche, du moins en ce qui concerne l'automatisation de leur traduction en les objets courants d'un langage de programmation.
Ces différents concepts seront détaillés au paragraphe 6.4.

6.2.1.2. Les concepts d'un langage de programmation

Les concepts relatifs aux structures de données sont bien entendu les types du langage, quelquefois appelés modes.
En ce qui concerne les structures de contrôle, une ambiguïté subsiste, car ce peut être soit les instructions du langage, soit leurs différentes utilisations possibles. La première hypothèse ne tient pas, et c'est sans doute ce qui explique l'échec de l'enseignement de la programmation par l'enseignement d'un langage.
Je ne citerai comme exemple que l'instruction IF logique de FORTRAN qui, sous sa forme la plus courante

IF (condition) GO TO étiquette

permet de décrire une action conditionnelle et un test de sortie de boucle.
C'est donc la seconde voie qu'il faut choisir, c'est-à-dire celle qui est équivalente à enseigner la traduction dans le langage des structures de contrôle du langage algorithmique (en général plusieurs traductions seront possibles).

S'ajouteront à ces deux groupes de concepts un troisième groupe lié à la structure du programme et à son exécution sous un certain système d'exploitation. Là encore des exemples seront fournis en 6.4. pour les langages PASCAL, LSE et BASIC.

6.2.2. *L'apprentissage en spirale*

L'idée n'est pas nouvelle : DEMARNE [49] l'utilisait déjà à propos de l'enseignement programmé lorsqu'il parlait, sans doute de manière un peu théâtrale, de programmes "en dents de scie, parallèles, en faisceaux, hélicoïdaux, en spirale ou en figures gauche". La définition de GLAESER nous semble plus réaliste [67] :

"Des notions particulièrement complexes ne peuvent pas être entièrement comprises en un seul instant, à un moment précis de la scolarité.

A leur propos, on peut opposer deux façons d'enseigner.

L'une d'elle consiste à exposer en une seule fois tout le contenu supposé de la matière à enseigner. Mais comme la compréhension complète n'est jamais acquise d'emblée, on procède chaque année à des révisions du même exposé, en misant sur l'efficacité de la répétition.

L'autre méthode s'appuie sur une analyse préalable des difficultés à surmonter, sur une description des seuils que l'élève doit franchir pour accéder à une compréhension complète. L'assimilation s'étend alors sur plusieurs années : à chaque reprise le niveau progresse ; on essaie de surmonter de nouveaux obstacles. C'est le principe de l'apprentissage en spirale".

Il parlait alors d'enseignement de l'analyse en mathématiques. Sur la figure 17, nous retrouvons la même stratégie dans l'enseignement du marketing à la British Airways [23] : la matière est séparée en un certain nombre de sujets, chaque sujet donnant lieu à des modules d'enseignement liés entre eux par des liens d'approfondissement. Dans la première moitié de la figure, seule la relation logique est notée ; on ne change de sujet que quand le précédent est complètement épuisé. Dans la seconde partie, on distingue trois niveaux de difficulté, et on ne passe au niveau supérieur que quand le niveau inférieur est terminé.

Enfin, les macromodens de PEUCHOT [46] peuvent servir à une telle stratégie.

Nous donnerons deux exemples de "spirale" sur des concepts simples de programmation, le premier concernant une structure de contrôle (l'itération) et le second une structure de données (le tableau).

Il est possible d'enseigner l'itération dans toute sa généralité, à savoir incluant l'arrêt au bout d'un nombre connu d'étapes, ou quand une certaine condition sur une suite qu'on calcule est vérifiée, ou quand le fichier sur lequel on lit les données se termine .... Les étudiants passeront ainsi un bon mois à itérer.

Il nous semble bien préférable de couper cette étude en trois et d'y intercaler, afin de laisser mûrir le concept, du traitement conditionnel, ou des procédures.

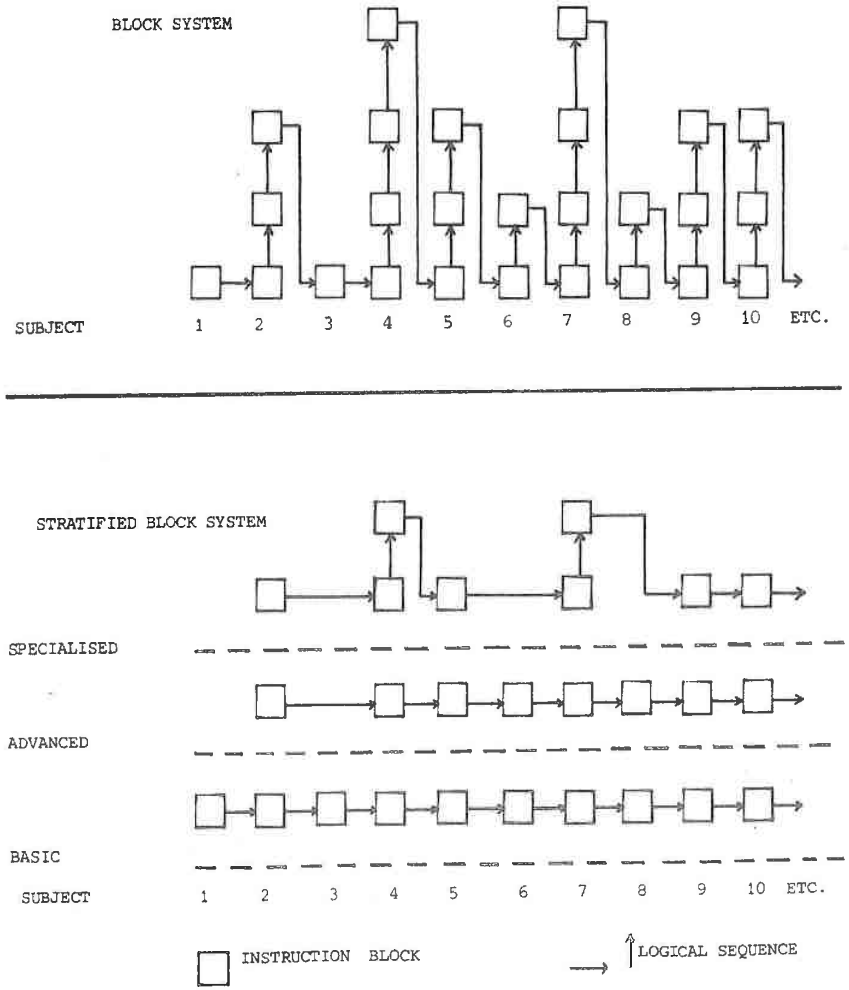


figure 17 : L'enseignement du marketing à la British Airways [23].

De même, en ce qui concerne les tableaux, si leur utilisation en lecture (consultation du prix d'achat d'un article) est immédiate et peut s'enseigner dès la première semaine, leur modification (mise à jour du nombre d'articles en stock) attendra une ou deux semaines et les tris sur les tableaux probablement deux bons mois.

6.2.3. *Langage ou pas langage ?*

L'enseignement de la programmation étant maintenant séparé en deux parties, l'enseignement de l'algorithmique et l'enseignement d'un langage, encore faut-il savoir si ces deux parties seront séquentielles ou parallèles, ce parallélisme faisant éventuellement l'objet d'un décalage plus ou moins grand. Les deux points de vue se défendent. Si on dispose d'un compilateur pour le langage algorithmique, ce qui est le cas à Nancy [83], on pourra reporter l'étude du langage tout à la fin. Par contre la disponibilité d'un langage conversationnel tel que BASIC ou LSE permettant un contact très proche entre l'étudiant et la machine poussera l'enseignant à faire suivre immédiatement l'étude du concept algorithmique par l'étude des concepts correspondants du langage de programmation.

6.3. L'APPRENTISSAGE PAR LA RESOLUTION DE PROBLEMES.

Dans l'état actuel de l'art, on ne dispose encore pas d'un système automatique d'élaboration de programmes, sinon ce problème d'enseigner la programmation ne se poserait pas ! En attendant, comment fait-on ? Paraphrasant PAPERT [150], je dirai qu'être un programmeur ne se définit pas plus comme "connaître" un ensemble de faits informatiques que être un poète ne se définit comme connaître un ensemble de faits linguistiques. Etre un programmeur (ou un mathématicien, ou un poète, ou un forgeron, ou un maçon, ...) signifie faire plus que connaître ou comprendre.

On enseigne donc la programmation en montrant sur des exemples comment on passe d'un problème à un programme et en donnant des problèmes aux étudiants. HOC a situé la programmation informatique par rapport à la situation générale de résolution de problème étudiée par les psychologues (figure 18). Il fait remarquer qu'en programmation l'élaboration d'une procédure utilise des évocations de procédure. En effet, dans un problème qui comporte une recherche de maximum dans un tableau, cette procédure sera insérée de façon quasi-automatique par l'étudiant qui l'aura déjà rencontrée plusieurs fois. Notons que dans la méthode de programmation déductive, le statut de plan de procédure est détenu par le guidage par les résultats, alors que les concepts de base ont plutôt le statut de contraintes syntaxiques.

Programmation informatique / résolution de problème	situation de production de programmes (expression)	situation de production de résultats (exécution)
situation d'élaboration de procédures	élaboration et expression d'une procédure	élaboration et exécution d'une procédure
situation d'évocation de procédures	simple expression d'une procédure	simple exécution d'une procédure

figure 18 : Rapports entre programmation informatique et résolution de problème selon HOC [78].

GORALSKI, dans une intéressante étude sur "le problème" dans les trois situations suivantes : école, vie pratique et recherche [68], distingue les étapes :

- formuler le problème
- résoudre le problème
- vérifier et valoriser la solution.

Il me semble intéressant de rapporter ici ce qu'il dit sur l'activité pratique, car elle décrit bien le travail des informaticiens dans une entreprise : (on appelle décidant le sujet formulant des problèmes et réalisateur celui qui les résoud)

"D1 le décidant estime qu'il a formulé le problème quand il a décidé de la nécessité d'un système de démarches où

1. l'effet de sa réalisation est la condition nécessaire de la satisfaction du besoin social aperçu
2. un réalisateur donné assure son exécution dans un temps défini, à l'aide des moyens distingués, dans des conditions existantes
3. ont été indiqués le déroulement espéré et l'effet de la réalisation ainsi que les manières de la valorisation"

C'est ce qu'on trouve, en général, dans ce qui est appelé le cahier des charges lors d'une étude d'informatisation d'une application.

"D2 le décidant donne l'ordre de la réalisation et désigne le réalisateur du problème car

- il a valorisé les réalisateurs en choisissant celui qu'il considère comme suffisamment efficace

- il connaît les conditionnements de l'exécution du problème
- il a estimé que le problème est réalisable car il y a eu des réalisations antérieures de tels systèmes de démarches, qu'elles sont équivalentes dans l'acceptation définie, englobant la comparaison des conditionnements, au système considéré
- il sait valoriser au moins un des effets de la réalisation
- il admet que la satisfaction des besoins importants du réalisateur sera l'un des effets de la solution du problème."

Il s'agit ici du choix du déroulement de l'informatisation (par société de services extérieure, ou par un service informatique propre à l'entreprise, ou par appel à un stagiaire, ...).

"R1. le réalisateur appelle processus de la solution du problème un système de démarches où

1. sa réalisation sera entreprise (le décidant en ayant décidé ainsi),
2. l'effet de l'exécution du système de démarches et les manières de la réalisation de l'effet seront valorisés par le décidant par rapport à la correspondance à l'effet et aux manières de la réalisation de l'effet espéré
3. le système peut être divisé en des démarches composantes de façon à ce que l'on puisse indiquer les réalisateurs et les moyens de la réalisation des démarches composantes, que les démarches composantes et l'ensemble du système de démarches puissent être réalisables parce que l'on connaît déjà les démarches modèles (qui, en l'occurence, seront répétées) et que l'effet global des démarches composantes ne soit pas inférieur à l'effet global espéré."

C'est la troisième assertion qui nous intéresse ici, dans la mesure où elle reflète bien l'activité habituellement appelée "analyse fonctionnelle". La citation se termine par un dernier paragraphe qui n'appelle aucun commentaire particulier. Tout aussi intéressantes, et plus proches des préoccupations de ce chapitre, sont les réflexions de GORALSKI sur la situation scolaire, dans laquelle le sujet formulant le problème est le maître, tandis que celui qui le résoud est l'élève :

"M1. le maître dit qu'il a formulé le problème quand il a indiqué une démarche de l'élève qui

1. sera exécutée (le plus souvent)
2. est la condition nécessaire de la transmission d'un certain savoir-faire
3. permet d'approcher son effet, distingué et espéré
4. permet de la valoriser dans son déroulement."

On peut donc considérer que la définition d'une méthode de programmation constitue, dans l'enseignement de cette discipline, la formulation du problème.

"M2. Le maître incite l'élève à résoudre le problème car

- il a valorisé les problèmes et a choisi celui qui lui a paru être le moyen efficace de la transmission du savoir-faire
- il sait comment valoriser la manière et les effets de la résolution
- il suppose que le savoir-apercevoir et le savoir-formuler le problème sera l'un des effets de la solution".

La troisième assertion est ici particulièrement intéressante, car c'est elle qui espère, par la méthode d'apprentissage retenue, la possibilité de transférer à des situations de la vie pratique ce qui a été fait pendant la phase d'apprentissage.

"E1. L'élève appelle résolution du problème une démarche qui

1. devra être réalisée dans un délai donné
2. fera que l'élève, lors de sa réalisation, ne pourra et ne devra imiter que ceux, parmi les actes du maître et ses actes personnels, qui ont été considérés par le maître comme utiles
3. lui permettra de connaître souvent le résultat espéré de sa réalisation, ce qui lui donnera la possibilité de valoriser la résolution.
4. fera que les effets de la démarche seront valorisés par le maître ou (et) par d'autres.

E2. L'élève entreprend la résolution du problème car il aperçoit que celui-ci peut être interprété comme un état de choses où il sera mauvais à un certain égard, s'il n'arrive pas à le résoudre et où il sera bien au même ou (et) à un autre égard, s'il arrive à le résoudre."

Dans ces deux derniers paragraphes interviennent les notions d'imitation et de motivation. Si la motivation est ici essentiellement externe (préparation d'un diplôme), encore que l'aspect "arriver à un programme correct" est important chez les étudiants en informatique, l'imitation devra être prise en charge par l'enseignement, surtout quand ce dernier est conçu sans intervention constante du maître, comme ce sera le cas dans l'enseignement assisté par ordinateur.

6.4. MACRO-ENSEIGNEMENT ASSISTE PAR ORDINATEUR DE LA PROGRAMMATION

L'enseignement de la programmation fournit au projet SATIRE un prototype intéressant pour plusieurs raisons :

- il nous donne la possibilité des tester l'outil "aide à la fabrication de cours" (5.4.1) sur une matière peu organisée et propre à accueillir des relations de nature très différente entre les concepts.

- il permet de tester la possibilité d'échanges constants entre l'utilisation de l'ordinateur comme instrument d'enseignement et comme calculateur.
- il est susceptible de fournir au groupe de recherche sur l'évaluation de la méthode de programmation déductive un grand nombre de programmes réalisés par des étudiants, ainsi que des traces de cheminement entre les différents concepts.
- il peut permettre à un public menant des études à temps partiel un apprentissage quasi autonome, ou une récupération d'absences en cours, ou des exercices répétés.

Il s'agit d'un macro-enseignement assisté par ordinateur au sens donné en 4.2. et pour lequel nous préciserons la nature des concepts, des leçons et des questions, ainsi que le choix momentanément effectué pour l'analyse des réponses.

6.4.1. *Les concepts*

Les concepts ont été choisis en fonction des considérations didactiques exposées au paragraphe 6.2. Ils concernent le langage algorithmique et les langages de programmation PASCAL, LSE et BASIC. On trouvera en annexe A de ce chapitre la liste des concepts de chaque sous-discipline, les numéros permettant de repérer leurs antécédents pour la relation d'antériorité directe $\mathcal{R}$. Des exmples de contenus des items correspondants à la version courante pour un public d'étudiants de 1ère année de département informatique d'IUT [10] sont donnés à l'annexe B de ce chapitre. Un des défauts couramment reprochés à la méthode de programmation déductive [134] est que les algorithmes finaux sont peu lisibles : en effet, ces algorithmes (voir un exemple à la figure 19) utilisent différentes plages de la feuille de papier ; lors de la construction de l'algorithme, le crayon se promène d'une plage à l'autre, et c'est cette promenade qui reflète la logique de la construction. Ayant donc le choix, pour les items, entre un support papier disponible dans un classeur à côté du terminal, et la transmission par le terminal du contenu de l'item, nous avons préféré la seconde solution, qui permet d'imprimer pour l'étudiant l'exemple que contient l'item dans l'ordre de sa conception, grâce aux fonctions micro-programmées (déplacement du papier et de la tête d'écriture) du terminal.

6.4.2. *Les leçons*

Cinq leçons du niveau d'une bonne initiation sont disponibles :

. *une leçon d'algorithmique seule.* La seule relation d'antériorité directe $\mathcal{R}$ ne permettant pas de prendre en compte le souci d'apprentissage en spirale, on lui a adjoint une relation $\mathcal{R}_{sp}$, les liaisons de $\mathcal{R}_{sp}$, qui figurent également dans l'annexe A, étant déduites de la progression que nous utilisons habituellement avec nos étudiants. Cependant, dans la programmation du contenu, $\mathcal{R}_{sp}$ ne joue pas nécessairement le même rôle que $\mathcal{R}$ (cf. chapitre 8).

- ALBUM (Edit) liste des gagnants de l'album - SAFARI (Edit) liste des personnes parmi lesquelles le sort désignera, si elles sont plus de deux, le gagnant du safari-photo - FINCAND (Bool) candidat de nom XXXX - NBGAG (Ent) nombre de concurrents ayant gagné un album		MODULE PRINCIPAL	- CARTE : lit la "carte" correspondant à un candidat, compare ses réponses (photos) à la réponse type, si elles sont toutes justes, imprime son nom, prénom, adresse (FINCAND, NBGAG, NBCAND) et mémorise les gagnants de l'album. - ICARTE : NOM, FINCAND, NBGAG, NBCAND. - GROSLOT : sélectionne les éventuels gagnants du safari-photo.
	2	ALBUM, NBGAG : ICARTE ; *jqd* FINCAND ≠ CARTE *avec* REPTYPE	
	3	SAFARI : *si* NBGAG ≠ 0 *alors* GROSLOT *sinon* SAFARI = *impr* 'Pas de gagnant au concours'	
	1	REPTYPE = *donnée*	
- NOM (chaîne) nom d'un candidat - GAGNANT (bool) réponses photos correctes - REP (chaîne) réponses (photos) - REPTYPE (chaîne) réponse type (photos)		CARTE	- EDITE-ET-GARDE : imprime nom, prénom, adresse
	9	FINCAND = (NOM = "XXXX")	
	8	NOM = *donnée*	
	7	ALBUM, NBGAG, PN, N, ADR, EST, D : *si* GAGNANT *alors* EDITE-ET-GARDE *avec* PRENOM, ADRESSE, $\overline{\text{NOM}}$, ESTIM, DATE	
	6	GAGNANT = REP = REPTYPE	
	2	REP = *donnée*	
	1	PRENOM, ADRESSE = *donnée*	
	3	NBCAND = NBCAND + 1	
	4	ESTIM = *donnée*	
	5	DATE = *donnée*	
- PRENOM (chaîne) - ADRESSE (chaîne) - ESTIM (ent) réponse à la question subsidiaire - DATE (ent) date d'envoi		EDITE-ET-GARDE	
	1	ALBUM = *impr* NOM, PRENOM, ADRESSE	
	2	NBGAG = $\overline{\text{NBGAG}}$ + 1	
	3	N(NBGAG) = $\overline{\text{NOM}}$	
	4	PN(NBGAG) = PRENOM	
	5	ADR(NBGAG) = ADRESSE	
	6	EST(NBGAG) = ESTIM	
	7	D(NBGAG) = DATE	
		ICARTE	
	1	NOM = *donnée*	
	2	FINCAND = (NOM = 'XXXX')	
	3	NBGAG = 0	
	4	NBCAND = 0	
- N (chaîne tab NBGAG) tableau des noms des gagnants de l'album - PN (chaîne tab NBGAG) prénoms - ADR (chaîne tab NBGAG) adresses		GROSLOT	- SUBSIDIAIRE : recherche celui ou ceux qui ont donné la meilleure réponse à la question subsidiaire
	1	*saut de page*	
	2	SAFARI : *si* NBGAG > 1 *alors* SUBSIDIAIRE *sinon* SAFARI = *impr* N(1), PN(1), ADR(1)	

figure 19 (début) : Exemple d'algorithme

- NBSUB (ent) nombre de concurrents ayant gagné un album et ayant donné la meilleure réponse à la question subsidiaire - NBIS, PNBIS, ADRBIS (chaîne tab NBSUB) voir ci-dessus pour les NBSUB concurrents		SUBSIDIAIRE	- DATE-D'ENVOI : recherche celui ou ceux qui ont la date d'envoi la plus précoce - CONSERVE : tableaux bis, NBSUB
	3	SAFARI : *si* NBSUB > 1 *alors* DATE-D'ENVOI *sinon* SAFARI = *impr* NBIS(1), PNBIS(1), ADRBIS(1)	
	2	NBSUB, NBIS, PNBIS, ADRBIS, DBIS : NBSUB = 0 *pour* I *dans* 1 → NBGAG *répéter* CONSERVE *avec* MEST	
	1	MEST : MEST = |EST(1) - NBCAND| ; *pour* I *dans* 2 → NBGAG *rep si* |EST(I) - NBCAND| < MEST *alors* MEST = |EST(I) - NBCAND| *fsi*	
- NBCAND (ent) : nombre de concurrents - MEST (ent) : meilleure estimation, à savoir minimum des différences entre les réponses et le nombre de concurrents - EST (ent tab NBGAG) : tableau des réponses à la question subsidiaire - D (ent tab NBGAG) : tableau des dates d'envoi		CONSERVE	
	1	*si* |EST(I) - NBCAND| = MEST *alors* NBSUB = NBSUB + 1 NBIS(NBSUB) = N(I) , PNBIS(NBSUB) = PN(I) , ADRBIS(NBSUB) = PN(I) , DBIS(NBSUB) = D(I) *fsi*	
		DATE-D'ENVOI	- EDITE : compare la date d'envoi à la date la plus précoce - POSTE
	3	SAFARI : *pour* I *dans* 1 → NBSUB *rep* EDITE *avec* MDAT	
- MAN (ent) année la plus précoce - NMOIS (ent) mois le plus précoce dans MAN - MJOUR (ent) jour le plus précoce dans NMOIS	2	MDAT = MAN + (NMOIS + MJOUR * 100) * 100	
	1	MAN, NMOIS, MJOUR : MAN = 78, NMOIS = 12, MJOUR = 31 ; *pour* I *dans* 1 → NBSUB *rep* POSTE	
- DBIS (ent tab NBSUB) tableau des dates d'envoi - MDAT (ent) date la plus précoce		EDITE	
	1	SAFARI : *si* DBIS(I) = MDAT *alors* SAFARI = *impr* NBIS(I), PNBIS(I), ADRBIS(I)	
- AN (ent) année dans DBIS - MOIS (ent) mois dans DBIS - JOUR (ent) jour dans DBIS		POSTE	- standard RESTE(x,y) reste et - standard DIV(x,y) quotient de la division entière de x par y
	4	*cas* AN < $\overline{MAN}$ *alors* MAN = AN ; MMOIS = MOIS ; MJOUR = JOUR AN = $\overline{MAN}$ *et* MOIS < $\overline{MMOIS}$ *alors* MMOIS = MOIS ; MJOUR = JOUR AN = $\overline{MAN}$ *et* MOIS = $\overline{MMOIS}$ *et* JOUR < $\overline{MJOUR}$ *alors* MJOUR = JOUR *sinon* rien *fcas*	
	1	AN = RESTE(RESTE(DBIS(I), 10000), 100)	
	2	MOIS = DIV (RESTE(DBIS(I), 10000), 100)	
	3	JOUR = DIV (DBIS(I), 10000)	

figure 19 (fin) : Exemple d'algorithme

. *trois leçons de programmation en LSE, PASCAL et BASIC.* Pour ces trois leçons, les concepts algorithmiques sont considérés comme prèrequis. L'apprentissage porte sur la syntaxe des instructions, les objets du langage et la traduction dans ces langages d'algorithmes conçus déductivement.

. *une leçon "intégrée" algorithmique + BASIC.* Pour cette leçon ont été utilisées les relations $\mathcal{R}$, $\mathcal{R}_{sp}$ et une relation $\mathcal{P}$ de priorité :

$$x \mathcal{P} y \Leftrightarrow y \text{ est présenté immédiatement après } x$$

qui traduit le fait que le concept BASIC correspondant à un concept algorithmique accompagne toujours ce dernier. Les liaisons de $\mathcal{P}$ figurent à l'annexe A.

6.4.3. *Les questions et l'analyse des réponses*

Ce qu'a dit SKEMP sur les mathématiques [185] s'applique assez bien à notre domaine : "la solution réussie d'un problème exige d'abord que les concepts nécessaires soient disponibles. Ensuite, que l'élève soit capable de choisir et modifier ces concepts selon les besoins. Si le premier essai est réussi, c'est que le problème était facile pour cet élève. La résolution des problèmes plus difficiles consiste essentiellement à utiliser les renseignements fournis par chaque essai infructueux pour diriger l'essai suivant, c'est-à-dire pour corriger les erreurs. Cela peut être fait au hasard ou de façon intelligente ... Supposons qu'on montre à un élève qui s'est trompé en quoi consiste sa faute, il corrige sa méthode sur ce point et parvient à résoudre le problème correctement".

Aussi, dans une situation idéale vers laquelle nous aimerions tendre, la procédure des questions prévue dans le mode tutoriel de SATIRE devrait être la suivante : l'étudiant construit son algorithme avec un outil conversationnel d'aide à la construction d'algorithmes déductifs. Cet outil aurait deux rôles :

- d'une part, enregistrer les erreurs commises par l'étudiant au cours de la construction, *sans le corriger*
- ensuite, reprendre avec l'étudiant la construction, *correcte cette fois*, en faisant un commentaire approprié sur ses erreurs.

Un tel outil, bien entendu, n'existe pas encore : le construire ferait presque l'objet d'une seconde thèse ! En l'attendant, le processus que nous avons retenu est le suivant :

- énoncé du problème
- l'étudiant résout son exercice, l'enregistre avec un éditeur de textes, le fait exécuter, récupère le résultat, modifie, ... autant de fois qu'il le désire
- une solution correcte lui est proposée
- il lui est demandé d'auto-évaluer les erreurs commises pour enregistrement, de manière conversationnelle. Il dispose pour ce faire de la liste des concepts mis en jeu par la question, et des différents messages reçus au cours des exécutions de son programme (erreurs à la compilation et à l'exécution) qui lui permettent de déterminer, parmi ces concepts, ceux dont une mauvaise maîtrise a provoqué les erreurs.

Les questions ont été collationnées et analysées par un groupe d'enseignants de l'IUT. On en trouvera quelques exemples à l'annexe D de ce chapitre.

ANNEXE 6A

Liste des concepts et relations entre les concepts

6.A.1. Cours d'algorithmique

Objectif terminal : être capable d'analyser tout problème simple, déterministe, non récursif, et ne nécessitant pas l'utilisation de structures de données autres que les nombres, les chaînes de caractères, les booléens et les tableaux.

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
1	schéma Problème → énoncé → algorithme → programme → résultats données ↗	71;72; 73;74		
2	définition de l'algorithme (ou analyse) ; nécessité de disposer d'un langage pour décrire les algorithmes	1		
3	Présentation de la méthode déductive : idées générales, disposition, un exemple	2		
4	Objets et opérations sur les objets ; objets simples et objets structurés. Notion de type	71;73	2	
5	Nombres réels	4;12	6	
6	Nombres entiers	4;12		
7	Chaînes de caractères	4;12	5	
8	Liste des définitions simples de la méthode déductive	3;4		
9	Les expressions ; les fonctions standard	8;12; 13		
10	Acquisition des données	13; 8;12; 71;73		

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
11	Impression de résultats	13; 8;12; 71;73		
12	Objets internes et externes dans la méthode déductive. Le type EDIT. Opérations sur les objets de type EDIT.	4;3; 71;73		
13	Notion d'identificateur	4;71; 12;72;73		
14	Lexique des identificateurs	3;13		
15	Problème séquentiel : définition et exemples	1	8;10; 11;14	
16	Enoncé déductif d'un problème séquentiel	3;14; 15		
17	Algorithme, ordonnancement	15		
18	Algorithme d'un problème séquentiel	16;17; 201;106		
19	Problème itératif : définition et exemples	1;15	18	
20	Problème itératif récurrent (calcul de u_{i+1} à partir de u_i, pas de cumul)	19		
21	Problème itératif pour lequel on connaît le cardinal de l'itération	19		
22	Définition itérative simple, sans impression	19		
23	Variables récurrentes (ancienne valeur)	20;13; 9		
24	Notion de module itéré, lexique des modules	22;19;		
25	Partie commune aux diverses exécutions d'un module itéré ; remontée des constantes d'itération.	24		

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
26	Problèmes itératifs récurrents avec cumul, initialisation	23;22; 24		
27	Module d'initialisation	26;24		
28	Itération et type EDIT	26;12	27	
29	Notion d'organigramme	17	28	
30	Organigramme d'un problème séquentiel	29;18		
31	Organigramme d'un problème itératif simple	17;21	30	
32	Problème conditionnel : définition et exemples	1	59	
33	Conditions, expressions booléennes, opérations non, et, ou	32;5; 6;7		
34	Problème à deux cas complémentaires	32;33		
35	Définition conditionnelle simple (pas d'impression)	34		
36	Modules conditionnels	35;24		
37	Type booléen	4;33		
38	Problème à plusieurs cas	34		
39	Définition conditionnelle par cas	38		
40	Module commun	36;39		

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
41	Conditionnelles et type EDIT	39;12	40	
42	Organigramme d'un problème conditionnel	29;32	40;41	
43	Itérations imbriquées	24;31	42	
44	Cas de plusieurs résultats définis par des itérations identiques : fusion d'itérations	22	43	
45	Diagramme de Karnaugh pour la simplification des conditions	39	55	
46	Conditions avec paramètre dans un ensemble fini	39	45	
47	Problèmes itératifs généraux : insuffisance de la forme POUR	21	46	
48	Itération sur un ensemble de départ de cardinal non connu	47		
49	Cas où le dernier élément est particularisé	48;37; 27		
50	Problème avec sentinelle ; lecture à l'avance	49		
51	Itération sans ensemble de données ou sur $\mathbb{N}$ (suites récurrentes avec arrêt sur précision)	47;37; 27		
52	Premier cas de 51	51;23		
53	Second cas de 51	51;23		
54	Organigramme d'un problème itératif général	29;49; 50;52;53		
55	Itérations simples par valeur décroissante ou avec un pas différent de un	22	44	

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
56	Tableau à une dimension	4;5; 6;7;	31	
57	Accès aux éléments d'un tableau ; utilisation dans des expressions	56;9		
58	Acquisition d'un tableau	10;56	57	
59	Impression d'un tableau	11;56	58	
60	Tableau à deux dimensions	56	54	
61	Tableau à un nombre quelconque de dimensions	60		
62	Recherche associative	57;53; 22	54	
63	Suites de tableaux définies par récurrence	57;58; 59;23; 50	54	
64	Recherche dichotomique	57;51	54	
65	Tri par sélection	63		
66	Tri par insertion	63	65	
67	Tri rapide	63	66	
68	Procédures : définition et exemple	1	67	
69	Procédures dans la méthode déductive	68;63		
70	Procédures : confusion entre arguments et résultats	69		

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
71	Ordinateur (prèrequis)			
72	Programme enregistré (prèrequis)			
73	Données et résultats d'un calcul (prèrequis)			
74	Langage de programmation (prèrequis)			
75	Compléments sur les conditionnelles	41		

6.A.2. Concepts de programmation indépendants du langage ; mode d'emploi de l'éditeur de textes, du système SIRIS 8 et de SNOOPY.

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
100	Notion d'instruction	71;72; 74		
101	Correspondance entre définitions de la méthode déductive et instructions	100		8
102	Ordre de traduction des définitions (pas de sous-module)	101		18
103	Ordre de traduction des définitions ; appel d'un sous-module ; introduction de commentaires	102		24
104	Présentation d'un travail (cartes de commandes dans le cas d'une exécution en traitement par lot)	100		
105	Utilisation de l'éditeur de textes pour fabriquer un fichier-programme et le modifier	71		

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
106	Soumission d'un travail au traitement par lot ; récupération du résultat	105		
107	Traduction du lexique : les déclarations		102	7
200	Présentation de SNOOPY, le compilateur de la méthode déductive	1		
201	Présentation d'un algorithme avec un seul module, cartes de commandes	200		
202	Présentation d'un algorithme avec plusieurs modules, sans procédures autres que les fonctions standard	201		
203	Présentation d'un algorithme avec procédures	202		
204	Fichiers-programmes relatifs à un algorithme	201;105		

6.A.3. Concepts relatifs au langage PASCAL sous SIRIS 8.

Seul un sous-ensemble de Pascal est étudié.

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
400	Présentation du langage PASCAL	74;100		
401	Structure du programme et commentaire	400		
402	Jeu de caractères, séparateurs, mots réservés, opérateurs, identificateurs, constantes, étiquettes ...	400		
403	Les types d'objets	402		
404	Types simples, scalaires, intervalles	403		
405	Affectation, expressions	402;404 107;418		
406	Instruction vide	400;100		
407	Instruction GO TO	400; 100		
408	Instruction composée	405; 103		
409	Instruction cas	408	410	
410	Instruction conditionnelle	431; 405; 408		
411	Instruction WHILE	405; 408	412	
412	Instruction REPEAT	405; 408	413	

n° du concept	description du concept	antécédents pour $\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
413	Instruction FOR	405; 408		
414	Fonctions standard	405		
415	Instruction PACK (dans le but de fabriquer du type ALFA)	420		
416	Cartes de commande sous SIRIS 7/SIRIS 8	104;106		
417	Liste des erreurs à la compilation	416		
418	Erreurs à l'exécution, post mortem dump	416	417	
419	Type ALFA	420;415		
420	Tableau à une dimension	404		
421	Tableau à plusieurs dimensions	420		
422	Instructions READ, READLN (objets simples)	403		
423	Instructions WRITE, WRITELN, formats (objets simples)	403		
424	Procédures : déclaration, corps, appel	405		
425	Paramètres d'une procédure	424		
426	Variables locales et variables globales	424	425	
427	Fonctions	424	426	

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
428	Lecture et impression des tableaux (dimension 1)	413; 420		
429	Impression du type ALFA	419; 423		
430	Lecture et impression des tableaux (dimension quelconque)	428		
431	Expressions booléennes	405		

6.A.4. Concepts relatifs au langages L.S.E. sous SIRIS 8.

Seul un sous-ensemble du langage L.S.E. est étudié.

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
300	Présentation du langage L.S.E.	74;100		
301	Notion de ligne en LSE (≠ instructions) ; commentaires	300		
302	Jeu de caractères, séparateurs, opérateurs, identificateurs, mots réservés, blancs	300		
303	Expressions, affectation	302;304; 107;306; 314		
304	Objets simples en LSE ; déclaration ; libération	331		
305	Instruction LIRE	304		
306	Instruction TERMINER	301		

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
307	Instruction AFFICHER sans format	304		
308	Instruction AFFICHER avec format	307		
309	Liste des erreurs à la compilation	310		
310	Cartes de commandes sous SIRIS 7/SIRIS 8	104 ; 106		
311	Sous-ensemble de fonctions standard	303		
312	Instruction FAIRE du premier type	303;103		
313	Ligne vide	301		
314	Erreurs à l'exécution, trace	309		
315	Instruction conditionnelle	330; 303;316; 317;318		
316	Instruction composée	303;103		
317	Procédures : déclaration, corps, appel	303		
318	Instruction ALLER EN	301		
319	Instruction FAIRE du 2e type	330	312	
320	Instruction FAIRE du 3e type	330	319	
321	Tableau à une dimension	304		

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
322	Lecture et affichage standard d'un tableau à une dimension	321;304; 307		
323	Tableau à deux dimensions	321		
324	Lecture et affichage standard d'un tableau à deux dimensions	323;304; 307		
325	Lecture et affichage "dirigés" des tableaux	312;324		
326	Paramètres d'une procédure	317		
327	Variables globales et variables locales	317	326	
328	Fonctions	317	327	
329	Fabrication de tableaux de chaînes de caractères	321		
330	Expressions booléennes	304		
331	Types d'objets	302		

6.A.5. Concepts relatifs au langage BASIC-C sous SIRIS 8.

Seul un sous-ensemble du langage est étudié.

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
500	Présentation du langage BASIC	74;100		18
501	Numéros de lignes, commentaires	500		
502	Jeu de caractères, mots réservés, opérateurs, identificateurs, ...	400		
503	Les types d'objets	502		
504	Objets simples	503		
505	Tableaux à une dimension (DIM)	504 (56)		
506	Tableaux à deux dimensions	505 (60)		
507	Expressions	504		
508	Fonctions standard	504		
509	Affectation (LET)	517;107 507; 522		
510	READ et DATA	504		
511	INPUT	504		
512	PRINT	504		

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
513	GO TO	501		
514	IF .... THEN	516; 513; 509;103		
515	FOR, NEXT	509; 103		
516	Sous-programmes	509		
517	STOP	501		
518	MATINPUT, MATREAD	506		
519	MATPRINT	506		
520	Sous-ensemble d'opérations sur les matrices	506		
521	Mise en oeuvre de BASIC sous SIRIS 7/SIRIS 8 LIST, DELETE, erreurs de syntaxe	71		
522	Exécution (GO TO, TRACE)	521		

6.A.6. Concepts relatifs au cours intégré algorithmique + BASIC-C sous SIRIS 8.

n° du concept	description du concept	antécédents pour		
		$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$
523	Traduction en BASIC d'un problème séquentiel	509;511; 512		
524	Traduction en BASIC d'un problème itératif (1)	515		28
525	Traduction en BASIC d'un problème conditionnel à deux cas complémentaires	514		36
526	Traduction en BASIC d'un problème conditionnel à plusieurs cas		525	41
527	Traduction en BASIC des itérations (1)	514		49
528	Traduction en BASIC des itérations (2)	514		50
529	Traduction en BASIC des itérations (3)	514		52
530	Traduction en BASIC des itérations (4)			53

ANNEXE 6B

Exemples de contenus d'items

Rappelons ici qu'il s'agit d'une version, dans le contexte "Première année de département informatique d'IUT", de certains des concepts de l'annexe 6A.
Les concepts présentés figurent dans le tableau récapitulatif ci-dessous :

n° du concept	description du concept	page
3	Présentation de la méthode déductive : idées générales, disposition, un exemple	155
25	Partie commune aux diverses exécutions d'un module itéré ; Remontée des constantes d'itération	156
105	Utilisation de l'éditeur de textes pour fabriquer un fichier-programme et le modifier	157
409	Instruction cas de PASCAL	158
315	Instruction conditionnelle de LSE	159,160
509	Affectation en BASIC	161
528	Traduction en BASIC des itérations (2)	162

Présentation de la méthode déductive

Nous avons montré précédemment l'intérêt, voire la nécessité, de disposer d'un *langage* pour décrire les algorithmes et d'une *méthode* pour passer d'un problème à un algorithme. Nous voulons ici vous présenter les premiers éléments d'une méthode (et du langage associé) que nous appelons la *méthode déductive*.

Un simple recours au bon sens suffit pour déterminer les outils que nous présentons. Les deux idées essentielles résultent d'une observation banale :

- l'ordre d'étude d'un problème est souvent différent de son ordre d'exécution (par exemple, un mathématicien cherchant à faire une démonstration jette des idées sur son papier en attaquant le problème tantôt par les hypothèses, tantôt par les conclusions présumées ; ce n'est que quand toutes les idées sont reliées entre elles qu'il rédige, de bout en bout et dans l'ordre des déductions logiques, la démonstration).
- au moment où on aborde un problème, ce que l'on connaît le mieux, c'est le résultat que l'on cherche à obtenir (par exemple, payer le personnel d'une entreprise) ; ce dont on a besoin pour obtenir ce résultat est inventorié en général après coup (dans notre exemple, la législation, le classement de chaque salarié, etc ...).

Ces constatations suggèrent la démarche suivante pour construire l'énoncé qui définit un problème :

a - partir du (ou des) résultat(s) final(s) pour définir les résultats intermédiaires et les données nécessaires à la résolution du problème.

b - à chaque instant, définir les résultats intermédiaires nécessaires au calcul.

Cette définition nous la ferons de deux manières différentes :

- de manière informelle, c'est-à-dire sous forme libre (en français), pour fixer les idées : c'est ce qui constitue une partie de ce qu'on appelle la *documentation* du programme.
- de manière formelle, c'est-à-dire par une définition précise écrite dans le langage associé à la méthode, définition utilisant d'autres résultats intermédiaires ou des données.

Un premier exemple nous permettra de comprendre les affirmations précédentes. On demande de trouver l'algorithme pour *calculer la somme à payer* pour un abonnement d'électricité de la série confort (et non imprimer une facture). Les caractéristiques de l'abonnement sont les suivantes : il y a deux tranches de consommation, la première à 48,15 centimes le kwh, la seconde à 15,04 centimes le kwh ; des frais fixes d'abonnement évalués à 25,20 F ; des taxes représentant 20,6 % du total (électricité + abonnement).

Quel est le résultat ? La somme nette à payer

Comment se calcule la somme nette à payer ? Somme nette à payer = somme brute + taxes

Comment définir la somme brute ? Somme brute = prix de l'électricité + montant de l'abonnement

Si on poursuit ce jeu de questions-réponses, on sera amené à écrire :

prix de l'électricité = prix de l'électricité en 1e tranche + prix de l'électricité en 2e tranche

prix de l'électricité en 1e tranche = quantité consommée en 1e tranche x 0,4815

prix de l'électricité en 2e tranche = quantité consommée en 2e tranche x 0,1504

$$\text{taxes} = \frac{\text{somme brute} \times 20{,}6}{100}$$

Les quantités consommées en 1e et 2e tranche sont les données dont on a besoin.

Notons qu'on pourrait prévoir une possibilité d'augmentation des tarifs et traiter ainsi un problème un peu plus général en considérant le montant de l'abonnement, le taux des taxes et les prix dans les deux tranches comme des données du problème.

Remontée des constantes d'itérations

Nous avons introduit précédemment les notions de module principal et module itéré. Nous les avons introduites en restant à un niveau *statique*, c'est-à-dire sans nous préoccuper du calcul qui réalisera cet algorithme en évaluant, à partir des données, les résultats.

Prenons un nouvel exemple : le gérant d'une cafeteria désire calculer combien lui doivent les clients qui ont consommé des cafés (les consommations sont notées dans un registre) pendant le mois écoulé. L'analyse du problème conduit à l'algorithme suivant :

- addition : calcul des notes des clients - NBCL(ent) : nombre de clients		Module principal	- ADD : calcul de l'addition d'un client
	2	addition : pour I dans 1 → NBCL rep ADD	
	1	NBCL = donnée	
- TOT(réel) : prix à payer par le client - NBCA(ent) : nombre de cafés consommés par le client - PRIX(réel) : prix d'un café		Module ADD	
	3	TOT = NBCA * PRIX	
	1	NBCA = donnée	
	2	PRIX = donnée	

Que se passe-t-il quand le calcul associé à cet algorithme est effectué sur un ensemble de données ? Le calcul correspondant au sous-module itéré est exécuté NBCL fois, et l'ordinateur cherchera à acquérir NBCL fois la donnée "prix d'un café" ; or cette donnée reste *constante* d'une exécution à l'autre (le prix du café n'est pas fait à la tête du client !). Autrement dit, cette définition est *commune* à toutes les exécutions du module itéré. Pour cette raison, elle sera *remontée* dans le module *appelant* ce module itéré, qui est ici le module principal. Il n'est pas nécessaire d'effectuer la même remontée au niveau du lexique des identificateurs. On obtient :

- addition : calcul des notes des clients - NBCL(ent) : nombre de clients		Module principal	- ADD : calcul de l'addition d'un client
	3	addition : pour I dans 1 → NBCL rep ADD avec PRIX	
	1	NBCL = donnée	
	2	PRIX = donnée	
- TOT(réel) : prix à payer par le client - NBCA(ent) : nombre de cafés consommés par le client - PRIX(réel) : prix d'un café		Module ADD	
	2	TOT = NBCA * PRIX	
	1	NBCA = donnée	

Notez que la numérotation des définitions a changé ; l'expression avec PRIX signifie que le module ADD utilise l'identificateur PRIX, facilitant ainsi l'ordonnancement du module principal.

Utilisation de l'éditeur de textes

Vous allez bientôt devoir écrire vous-mêmes des algorithmes ou des programmes, et nous vous demanderons d'en vérifier la validité en les soumettant à la sagacité de différents logiciels disponibles sur l'ordinateur IRIS 80 : SNOOPY, le compilateur de la méthode déductive, les compilateurs PASCAL ou LSE, l'interpréteur BASIC. Pour cela, il faut que vous enregistriez dans la mémoire de l'ordinateur ce que vous aurez écrit sur votre feuille de papier. Nous appellerons cela un *fichier-programme*. Il existe un logiciel qui permet de constituer ces fichiers programmes et de les modifier : un tel logiciel s'appelle un éditeur de textes et son exécution est lancée en tapant EDIT sur la console, de même que vous avez tapé SATIRE pour entamer ce dialogue.

Attention : il faudra auparavant quitter SATIRE en tapant PAUSE (après avoir reçu une * du système), et attendre de recevoir !

Chaque fichier-programme est repéré par un nom qui lui est donné à la création. Les lignes d'un fichier-programme sont numérotées avec un nombre décimal. Nous allons vous montrer comment on crée un fichier-programme (ici ce sera un texte de poème : vous verrez plus tard ce qu'il faut faire pour un algorithme ou un programme). Selon la convention habituelle, ce que l'ordinateur envoie est souligné.

```
!EDIT
V1503 062P00 00/002/102
*B/FICH-1/
00001.002:LA CIGALE AYANT DANSE TOUT L'ETE,
00002.002:SETROUVA FORT DEPOURVUE QUAND LA BISE FUT VENUE.
00003.002:AH! COMBIEN DE MARIS, COMBIEN DE CAPITAINES,
00004.002:PAS UN SEUL PETIT MORCEAU DE MOUCHE OU DE VERMISSEAU
00005.002:CHEZ LA FOURMI SA VOISINE
00006.002:
00001.002*
```

(B signifie BUILD (construire)
FICH-1 est le nom du fichier)

Si on a quitté l'éditeur de textes et qu'on y revient ensuite, il nous indique le nom du "fichier-courant" :

```
!EDIT
V1503 062P00 00/002/102
CURRENT FILE : FICH-1
00001.002*
```

Si c'est sur ce fichier que l'on veut travailler, c'est bien. Sinon, il faut rechercher le fichier en question :

```
!EDIT
V1503 062P00 00/002/102
CURRENT FILE : ESSAI
00001.202*F/FICH-1/
00001.202*
```

(F signifie FIND (chercher))

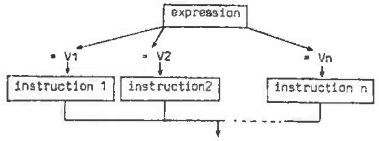
Maintenant, que peut-on faire sur ce fichier ? Voici les commandes de listage (L) d'un groupe de lignes, suppression (D) d'une ligne, insertion (I) d'une ligne, modification (S) d'une ligne, qui nous ont permis d'arriver à du LA FONTAINE à peu près correct. Pour avoir davantage de détails sur l'utilisation de l'éditeur de textes (ce sera utile dans l'avancement de votre travail), vous vous reporterez à la notice qui se trouve à proximité du terminal.

```
00001.000* 3D
00001 LINES DELETED
00004.000* 4.5I
00004.500: ELLE ALLA CRIER FAMINE
00004.500* 1S/DANSE/CHANTE/
00001.000  LA CIGALE AYANT CHANTE TOUT L'ETE
00001.000* 1,5L
00001.000  LA CIGALE AYANT CHANTE TOUT L'ETE
00002.000  SE TROUVA FORT DEPOURVUE QUAND LA BISE FUT VENUE.
00004.000  PAS UN SEUL PETIT MORCEAU DE MOUCHE OU DE VERMISSEAU !
00004.500  ELLE ALLA CRIER FAMINE
00005.000  CHEZ LA FOURMI SA VOISINE
00005.000*
EOF HIT
00005.000*
```

Si vous vous trompez en tapant les commandes, l'éditeur vous le signalera et attendra que vous la retapiez. Bon courage ! Si vous voulez, vous pouvez quitter SATIRE maintenant pour vous amuser un peu avec l'éditeur. Pour quitter l'éditeur, il suffit de taper END après l'*. En revenant dans SATIRE, vous reprendrez à l'endroit de l'interruption.

Instruction cas de PASCAL

Il existe dans le langage PASCAL une instruction permettant d'exprimer des actions conditionnées par une expression prenant ses valeurs dans un ensemble fini de valeurs ; elle correspond au schéma suivant :



(Les Vi sont appelées étiquettes du cas)

(instruction i peut être vide, simple ou composée)

Les étiquettes de cas doivent être toutes des constantes du même type, ce type ne pouvant être que entier, caractère ou booléen. L'ordre des éléments du cas n'est pas significatif. L'expression doit avoir le même type que les étiquettes du cas, sinon une erreur est détectée à la compilation. Lors de l'exécution de cet "aiguillage", l'expression est évaluée. Deux cas peuvent se produire :

- soit la valeur de l'expression est égale à une des étiquettes de cas : l'instruction correspondante est exécutée, et l'instruction cas est terminée
- sinon : il s'agit d'une erreur qui peut même conduire à interrompre l'exécution du programme tout entier.

Syntaxe de l'instruction :

```
<instruction cas> ::= CASE <expression> OF <élément cas>
                                    {,<élément cas>} * END
<élément cas> ::= <étiquette de cas>{,<étiquette de cas>}* : <instruction>
<étiquette de cas> ::= <constante>
```

Exemple :

```
VAR I : INTEGER ;
      :
CASE 2*I+1 OF

    -1,-3 : X := 0 ;
    1,3,5,7 : X := X+1 ;
    17,19,21 : X := X-1 ;
```

Exemple d'utilisation :

L'instruction cas de PASCAL est très utile pour traduire les définitions conditionnelles par cas de la méthode déductive, quand les conditions portent sur une donnée codée (sexe, catégorie de personnel, etc...). Voici un programme PASCAL pour un problème (très simplifié) de paiement d'heures complémentaires dont l'algorithme a été construit dans le message relatif au concept "conditions avec paramètre dans un ensemble fini" de la banque de concepts de SATIRE (vous pouvez éventuellement l'obtenir par le mode documentaire) :

```
PROGRAM VACATAIRES;
CONST  CEC1='R';CEC2='E';CEC3='L';CCAT1=1;CCAT2=2;CCAT3=3;SS1=1;SS2=2;TX=0.004;
       TH1=80.0;TH2=100.0;TH3=120.0;
TYPE   CHAINE=ARRAY[1..12]OF CHAR;
VAR    MOIS,NOM:CHAINE;NBH,J,I,N,CCAT,SS:INTEGER;SB,SN,RET,TH:REAL;CEC:CHAR;
BEGIN
       FOR I:=1 TO 12 DO BEGIN READ(MOIS[I])END;READLN(N);
       FOR I:=1 TO N DO % BULLETIN %
          BEGIN READLN(CEC);FOR J:=1 TO 12 DO BEGIN READ(NOM[J])END;
          READLN(CCAT,NBH,SS);
          WRITE('MOIS DE ');FOR J:=1 TO 12 DO BEGIN WRITE(MOIS[J])END;WRITELN;
          CASE CEC OF
               CEC1 : WRITE('MR : ');
               CEC2 : WRITE('ME : ');
               CEC3 : WRITE('MELLE : ')
          END;FOR J:=1 TO 12 DO BEGIN WRITE(NOM[J])END;WRITELN;
          WRITE('CATEGORIE : ');
          CASE CCAT OF
               CCAT1 : BEGIN TH:=TH1;WRITE('ASSISTANT') END;
               CCAT2 : BEGIN TH:=TH2;WRITE('MAITRE DE CONFERENCES') END;
               CCAT3 : BEGIN TH:=TH3;WRITE('PROFESSEUR') END
          END;WRITELN;
          WRITELN ('NOMBRE D''HEURES : ',NBH);
          SB:=TH*NBH;WRITELN('SALAIRE BRUT : ',SB:8:2);WRITE('RETENUES : ');
          CASE SS OF
               SS1 : BEGIN RET:=SB*TX;WRITELN(RET:8:2);SN:=SB-RET END;
               SS2 : BEGIN SN:=SB;WRITELN('NEANT') END
          END;
          WRITELN('SALAIRE NET : ',SN:8:2);
          WRITELN
          END % BULLETIN %
END.
```


Instruction conditionnelle de L.S.E.

Le langage LSE, à cause de sa structure de lignes numérotées, ne se prête pas aussi bien que d'autres langages (PASCAL, PL/1) à la traduction des définitions conditionnelles de la méthode déductive. Si on veut s'en tenir à la règle - sage! - de ne pas utiliser trop d'instructions ALLER EN car elles rendent le programme peu lisible, on choisira, en fonction de la forme de la définition, l'une ou l'autre des traductions suivantes :

1. Utilisation de procédures internes sans paramètres

1.1. définition à deux cas complémentaires

définition algorithmique

si cond *alors* mod1 *sinon* mod2

programme L.S.E.

```
:
N1  SI cond ALORS &MOD1( ) SINON &MOD2( )
:
N2  TERMINER
N3  PROCEDURE &MOD( )
:       traduction de mod1
N4  RETOUR
N5  PROCEDURE &MOD2( )
:       traduction de mod2
N6  RETOUR
```

1.2. définition conditionnelle par cas

si cond1 *alors* mod1
si cond2 *alors* mod2
:
cond6 *alors* mod6

(le choix de 6 est arbitraire !)

```
:
N1  SI cond1 ALORS &MOD1( )
N2  SI cond2 ALORS &MOD2( )
:
N3  SI cond6 ALORS &MOD6( )
N4  suite du programme
:
N5  TERMINER
N6  PROCEDURE &MOD1( )
:       traduction de mod1
N7  RETOUR EN N4
N8  PROCEDURE &MOD2( )
:       traduction de mod2
N9  RETOUR EN N4

etc jusqu'à &MOD6( )
```

Notez bien que :

- cond2, ... cond6 peuvent devenir dans le programme des conditions plus simples ; si vous voulez des compléments sur ce point, reportez vous, grâce au mode documentaire, au concept "compléments sur les conditionnelles" de la banque de concepts de SATIRE
- si l'union des conditions est vraie, il est inutile de tester cond6 et la ligne N3 devient
  N3 &MOD6( )
- la mention de la ligne de retour dans les instructions RETOUR des procédures n'a d'autre but que d'améliorer l'efficacité du programme : dès l'instant où une condition est vraie, ce n'est pas la peine de tester les autres, puisqu'elles s'excluent deux à deux.

2. Traduction "littérale"

Quand les modules mod1, mod2, ... mod6 ne contiennent que deux définitions (ou une seule : dans ce cas on n'a même pas introduit de modules!), on hésite à se servir des procédures.
On peut alors faire une traduction "littérale" à condition de ne pas dépasser les quatre lignes physiques automisées pour une ligne LSE par le compilateur IRIS 80 :

N1 SI cond1 ALORS traduction de mod1 SINON traduction de mod2

N2 SI cond1 ALORS traduction de mod1 SINON SI cond2 ALORS traduction de mod2 SINON SI cond3 ALORS traduction de mod3 etc...

.../...

Notez bien que :

- la remarque faite ci-dessus sur cond1, ..., cond6 s'applique ici
- de même, si l'union des conditions est vraie, la ligne N2 se terminera par .... SINON traduction de mod6
- l'appel des procédures (exemple donné plus haut) peut se faire sur une seule ligne, comme ici ; cela dispense de l'indication de la ligne de retour. On en verra un exemple ci-après.

3. Exemples d'utilisation

Les programmes suivants sont la traduction des algorithmes construits dans les concepts "définition conditionnelle simple" et "définition conditionnnelle par cas" de la banque de concepts de SATIRE, auxquels vous pouvez vous reporter grâce au mode documentaire.

définition conditionnelle simple	avec procédure	1 : LIRE A,B 5 : SI A<B ALORS &MAXB() SINON &MAXA() 10 : AFFICHER 'LE MAXIMUM DE ',A,' et ',B,' EST ',MAX 15 : TERMINER 20 : PROCEDURE &MAXA() 25 : MAX<- A;RETOUR 30 : PROCEDURE &MAXB() 35 : MAX<-B;RETOUR
	sans procédure	3 : LIRE A,B 10 : SI A<B ALORS MAX<-B SINON MAX<-A 15 : AFFICHER 'LE MAXIMUM DE ',A,' et ',B,' EST ',MAX 20 : TERMINER
définition conditionnelle par cas	avec procédure	1 : LIRE SB,NH 5 : SI NH<= 40 ALORS &SAL1() SINON SI NH = 48 ALORS &SAL2() SINON &SAL3() 15 : AFFICHER 'LE SALAIRE CORRESPONDANT A ',NH' HEURES SUR LA BASE DE ',SH,' FRANCS DE L''HEURE EST DE ',SAL,' FRANCS' 20 : TERMINER 25 : PROCEDURE &SAL1() 30 : SAL<-SH*NH;RETOUR 35 : PROCEDURE &SAL2() 40 : SAL<-SH*(NH+0.5*(NH-40));RETOUR 45 : PROCEDURE &SAL3() 50 : SAL<-SH*(NH+0.5*8+0.75*(NH-48));RETOUR
	sans procédure	1 : LIRE SH,NH 5 : SI NH<= 40 ALORS SAL<-SH*NH SINON SI NH<= 48 ALORS SAL<-SH*(NH+0.5*(NH-40)) SINON SAL<-SH*(NH+0.5*8+0.75*(NH-48)) 15 : AFFICHER 'LE SALAIRE CORRESPONDANT A ',NH,' HEURES SUR LA BASE DE ',SH,' FRANCS DE L''HEURE EST DE ',SAL,' FRANCS' 20 : TERMINER

Affectations en BASIC - Instruction LET

L'affectation est une instruction qui permet de donner une valeur à un objet simple repéré par un identificateur. Cette valeur peut être une constante, ou celle d'un autre objet repéré par un identificateur, ou le résultat de l'évaluation d'une expression au moment de l'exécution de l'instruction.

Syntaxe

<affectation> ::= <numéro ligne>{LET}<affectation simple>{,<affectation simple>}*

<affectation simple> ::= <variable>{,<variable>} * =<expression>

Sémantique

L'ensemble des variables et des constantes relatives à un signe égal (=) doivent être de même type, mais les différentes affectations simples peuvent être de types différents.

L'expression à droite du signe = est d'abord calculée et sa valeur est affectée aux variables en commençant par celle qui se trouve le plus à gauche du signe =. Quand il y a plusieurs affectations simples, elles sont réalisées de la gauche vers la droite.

Exemples

```
10  A = 1
20  LET B,C,D = 17
40  LET A$ = 'ALPHA'
50  B$ = A$
60  A = 1,B,C = 2
70  LET D = 0, B$, C$ = 'LETTRE'
80  E = A + B
90  A = A + 1
```

Traduction en BASIC des itérations (2)

Nous venons d'analyser un problème itératif pour lequel l'arrêt de l'itération dépend d'une valeur "sentinelle" qui termine l'ensemble des données. Nous allons voir comment l'algorithme que nous avons construit peut être programmé en BASIC (Nous disons *peut*, car bien entendu la solution n'est pas unique, mais nous vous rappelons que pour notre part nous souhaitons que la traduction algorithme → programme soit effectuée de façon systématique).
Nous allons utiliser pour cette traduction les instructions IF .... THEN et GO TO.
Voici d'abord le liste du programme, les explications suivront.

```
10 PRINT 'RESULTATS'
20 REM D/ELEVINI/
30 N2,T=0
40 INPUT N$
50 REM F/ELEVINI/
60 IF N$='XYZ' THEN 130
65 REM D/ELEVE/
70 INPUT N1
80 PRINT N$,N1
90 N2=N2+1
100 T=T+N1
110 INPUT N$
120 GO TO 60
125 REM F/ELEVE/
130 M=T/N2
140 PRINT 'MOYENNE : ',M
150 STOP
```

Lexique des identificateurs	
MOY	M
TOT	T
N	N2
NOM	N$
NOTE	N1

La définition "*pour* nom, note *dans donnée lim* nom = 'XYZ' *excl*" ne prévoit pas, d'autre part l'acquisition de nom et note dans les données, d'autre part la manière dont la fin des données sera détectée. Parlons simultanément des deux problèmes : nous avons décidé de lire un nom et de nous demander si ce nom était ou non 'XYZ'. Si non, le traitement du module élève est effectué, y compris la lecture de la note ; si oui, on retourne aux dernières instructions du module principal. Notons que nous faisons une lecture à l'avance : ceci nous permet de traiter le cas où il n'y aurait aucun élève ! En conséquence, l'instruction INPUT N$ figure à deux endroits dans le programme.

ANNEXE 6C

Exemple de questions posées dans une leçon

Ces exemples sont issus du cours d'algorithmique.

Q1 : On donne deux noms masculins et un verbe du premier groupe à l'infinitif. Ecrire l'algorithme qui imprime une phrase avec le premier nom comme sujet, le verbe au présent et le deuxième nom comme complément, et imprime également la forme passive de cette phrase (exemple : le train blanchit le lion, le lion est blanchi par le train).

Concept après lequel la question est posée : algorithme d'un problème séquentiel (18).
Composants de la question : 7,9,10,11,12,14,16,17,18.

Q2 : Les quatre coefficients entiers a_0, a_1, a_2, a_3 et une valeur réelle v étant donnés, écrire un algorithme qui imprime la valeur du polynôme

$$a_0 + a_1 + a_2 x^2 + a_3 x^3$$

pour x=v.
On utilisera la forme canonique ci-dessus et la forme suivante

$$a_0 + x(a_1 + x(a_2 + xa_3))$$

Concept après lequel la question est posée : algorithme d'un problème séquentiel (18).
Composants de la question : 5,9,10,11,14,16,17,18.

Q3 : Un examen comporte trois épreuves. Des coefficients différents sont attribués aux épreuves, et sont donnés. Pour chaque candidat, on dispose du numéro d'académie, du numéro d'établissement, du numéro de candidat et de son nom, de ses trois notes. Ecrire un algorithme qui imprime un tableau comportant une entête et une ligne par candidat comportant les données ci-dessus et le total obtenu à l'épreuve. L'impression se terminera par quatre étoiles au milieu de la page. Le nombre de candidats est donné.

Concept après lequel la question est posée : itérations et type edit (28).
Composants de la question : 5,7,9,10,11,12,14,22,24,25.

Q4 : Ecrire un algorithme qui imprime les n premiers nombres de FIBONNACI

$$\begin{cases} u_0 = 0 \\ u_1 = 1 \\ u_n = u_{n-1} + u_{n-2} \end{cases}$$

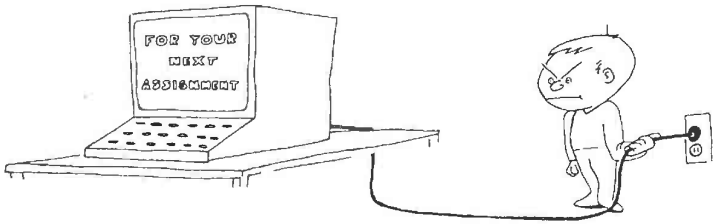
Concept après lequel la question est posée : itérations et type edit (28).

Composants de la question : 6,11,14,22,23,24,27,28.

Q5 : Ecrire un algorithme qui imprime tous les diviseurs d'un entier donné.

Concept après lequel la question est posée : conditionnelles et type edit (41).

Composants de la question : 6,9,11,14,22,24,28,35,37,41.



CHAPITRE 7 : Interface de SATIRE avec le système SIRIS 8 et le SGBD Socrate II. Insuffisances et propositions.

7.1. CONSIDERATIONS DIDACTIQUES ET CONSEQUENCES TECHNIQUES.

Au paragraphe 2.4.2, nous avons insisté, grâce à un inventaire emprunté à FAFIOTTE [59], sur une propriété que devait avoir tout système d'E.A.O. de type tutoriel : être "ouvert". Par système "clos", il faut entendre que quand l'ordinateur est utilisé pour l'E.A.O., il est inutilisable par l'étudiant pour une autre fonction : calcul, programmation, simulation, ... Les concepteurs des systèmes ont donc été contraints, quand la pédagogie l'exigeait, de bricoler, à l'intérieur du système, une "machine de bureau", un "mini-langage de programmation", on en a un exemple à l'O.P.E. [145].

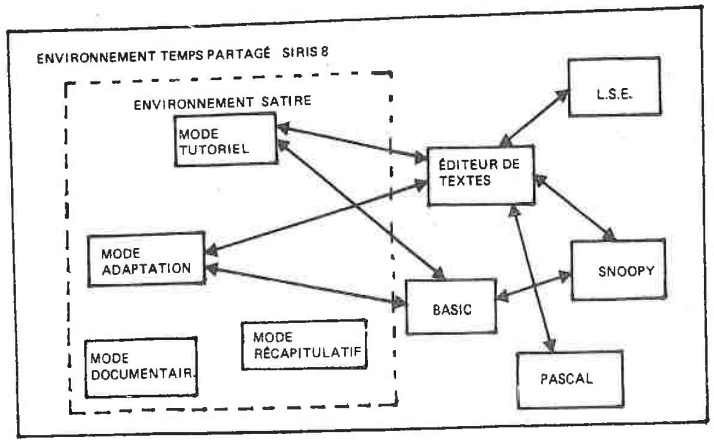
Notre démarche a été tout à fait différente : partant du principe que chaque discipline possède une didactique particulière dans laquelle l'utilisation possible de l'ordinateur pour transmettre les concepts ou résoudre les problèmes est différente de celle des autres disciplines, il était impératif de pouvoir quitter le système à tout moment pour utiliser d'autres logiciels.

Le physicien voudra en effet lancer des programmes de simulation, le gestionnaire exécuter des jeux d'entreprise, le psychologue des programmes d'analyse de données.

En ce qui concerne l'enseignement de la programmation, nous voulons qu'un étudiant puisse s'exercer à écrire des algorithmes et des programmes. Nous disposons sur l'ordinateur des logiciels suivants :

- SNOOPY, le compilateur de la méthode de programmation déductive [83]
- BASIC, langage conversationnel [9]
- PASCAL et LSE, langages compilés, qui nécessitent donc l'enregistrement du programme et des données dans un fichier grâce à un EDITEUR DE TEXTES, également disponible [7, 81].

Ces logiciels étant utilisables dans l'environnement du temps partagé du système SIRIS 8 [136], le programme de dialogue ordinateur-étudiant du projet SATIRE devait donc se situer dans cet environnement, selon le schéma suivant :



En outre, l'utilisation de l'ordinateur pour SATIRE restant très marginale, il n'était pas question de se situer dans un autre environnement que celui du système d'exploitation normal.

7.2. LE SYSTEME DE GESTION DE BASES DE DONNEES SOCRATE II [188].

Nous avons vu en 5.2. l'intérêt d'utiliser une base de données pour stocker les différents objets du système. Nous avons donc décidé d'implanter la base de données du projet SATIRE sous le SGBD SOCRATE II. On trouvera à l'annexe A de ce chapitre la description "théorique" de la structure de la base, c'est-à-dire sans les spécifications de détails de représentation (nombre de réalisations des entités, taille des objets physiques correspondant aux différentes caractéristiques) et incluant la totalité des objets didactiques recensés dans les chapitres 3 et 4. La base effectivement réalisée, qui comporte un certain nombre de simplifications et les choix de représentation est présentée au chapitre 8. Nous souhaitons faire ici un rapide bilan de l'adaptation du SGBD à cette application particulière.

7.2.1. *Structure logique de la base.*

Les ensembles d'objets recensés au paragraphe 5.3. sont réalisés par des ENTITES, la notion d'ENTITE IMBRIQUEE rendant compte de l'idée de démultiplication. A la figure 20, on a fait figurer les relations logiques entre les diverses entités. La

double liaison correspond à l'imbrication. Les liaisons en pointillés sont, pour une réalisation d'entité, un pointeur vers une réalisation d'une autre entité : il s'agit donc d'une caractéristique REFERENCE qui sera, selon les cas, simple, associée à une référence, vers entité ou associée à une relation de type chaîne. Une caractéristique de type INVERSE permet de repérer un sous-ensemble des réalisations d'une entité, comme c'est le cas des prérequis de la base demandés par les étudiants.

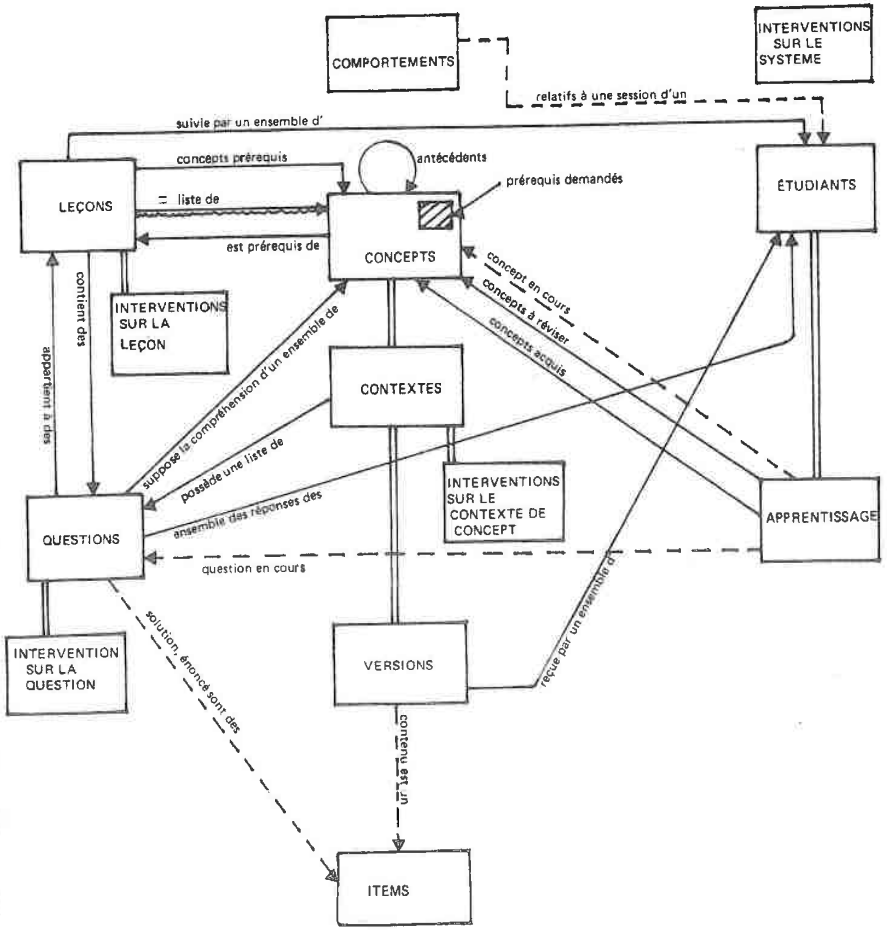


figure 20. Entités et relations entre les entités.

Les autres relations associent, à une réalisation d'entité E1, un ensemble de réalisations d'une autre entité E2 (remarquons que dans le cas leçon = liste de concepts, il s'agit d'un multi-ensemble dont la représentation est nécessairement un tableau). Il semble donc, au premier abord, qu'on puisse utiliser une caractéristique CHAINE ou LISTE, selon le cardinal de l'ensemble associé, laissant ainsi au SGBD le soin de gérer les pointeurs. Mais cela ne convient pour aucune de nos relations : en effet, si on appelle E1 l'entité PERE, et E2 l'entité FILLE, (et en vertu d'une jurisprudence selon laquelle une fille ne saurait avoir deux pères !), une même réalisation de E2 ne pourra se trouver sur deux chaînes (ou listes) différentes : en aucun cas les leçons ne pourront se partager les questions ou les concepts, ni les questions les concepts...

SOCRATE II, même s'il baptise les liens entre entités du nom pompeux de relations, reste donc dans une bonne tradition hiérarchique, les subordonnés ayant juste le droit ... de se donner quelquefois la main. Une des conséquences pour nous a été la nécessité d'exprimer toutes ces relations par des tableaux contenant des numéros de réalisation, ce qui a considérablement alourdi la programmation et ôte de la lisibilité à la structure logique.

Un autre exemple : pour un contexte de concept, il faut connaître l'ensemble des questions qui seront posées après son étude. Ceci n'est valable que pour une leçon donnée, puisque ces questions mettent en relation le concept avec les concepts antérieurement étudiés, et que ces derniers varient d'une leçon à l'autre. On aimerait donc disposer d'une caractéristique paramétrée par le numéro de la leçon, ce qui serait possible si le SGBD mettait à la disposition de l'utilisateur des constructeurs de caractéristiques comparables aux constructeurs de modes d'ALGOL 68 [66]. Ce n'est pas le cas, on en est donc réduit à dupliquer la caractéristique (grâce à un tableau) et à la transformer en bloc, ce bloc incluant comme caractéristiques simples le numéro de la leçon et la caractéristique initiale, ce qui alourdit la requête d'accès et la structure.

Il nous semble, sans que nous l'ayons approfondi particulièrement, que les concepts de structure et d'unités de sélection de PIVOINES [47] nous auraient permis une description plus souple de l'information manipulée par le projet SATIRE. Si l'étude théorique pouvait présenter de l'intérêt, l'application pratique était malheureusement impossible.

7.2.2. *Mise en oeuvre de SOCRATE II.*

Il y a deux modes d'accès aux informations contenues dans la base : depuis un programme écrit en langage évolué (la base se comporte alors comme un simple fichier à accès direct dont on atteint les éléments en se guidant sur la structure et en

utilisant des sous-programmes écrits dans le langage de requêtes) ou de manière conversationnelle, à partir de terminaux, en utilisant uniquement le langage de requêtes.

Le deuxième mode est en principe destiné à des non-informaticiens. Le langage de requêtes est agréable et relativement proche du langage naturel. Dans la mesure où les questions sont très répétitives, un macro-générateur permet d'abréger leur rédaction. Il n'était donc pas complètement irréaliste de penser que les fonctions de SATIRE relatives à l'enseignant pouvaient être écrites dans le mode conversationnel : on fait alors simplement l'hypothèse que le langage de requêtes n'est pas plus difficile à apprendre qu'un langage auteur. Seule la conception assistée d'une leçon est nécessairement un programme écrit dans un langage évolué, même si elle s'exécute de façon conversationnelle.

Si on considère maintenant le dialogue de l'étudiant avec ordinateur, qui est lui aussi nécessairement piloté par un programme, il se pose le problème de l'accès multiple à la base de données. Or, si ce problème est résolu dans le mode conversationnel multi-consoles (encore appelé mode transactionnel), il ne l'est pas dans un environnement de temps partagé ou de multiprogrammation : nous voilà donc réduits à faire travailler un seul étudiant à la fois, ce qui limite considérablement l'intérêt de l'enseignement assisté par ordinateur.

7.2.3. *Nature des items et inexistence d'une caractéristique adéquate.*

On a vu au § 2.4.2 que tout système "moderne" d'E.A.O. se devait d'enrichir le contenu informationnel des items, et que le système PLATO, entre autres, le permet. Les systèmes de gestion de bases de données, et SOCRATE II n'échappe pas à cette règle, ont été écrits pour mémoriser les données que l'on trouve couramment dans les fichiers destinés à la gestion des entreprises, c'est-à-dire essentiellement des données numériques ou textuelles : pas des programmes. On peut en déduire qu'actuellement le système SATIRE ne peut, sans artifice (un tel artifice est possible mais demande une coopération de l'étudiant, c'est-à-dire ôte de la transparence au système), avoir des contenus d'items autres que textuels.

Ceci n'était pas pour nous une préoccupation immédiate, puisque notre intention, dans l'enseignement assisté de la programmation, est de mémoriser des textes, ceux-ci incluant éventuellement des caractères correspondant aux fonctions programmables du terminal telles que déplacement du papier ou de la tête d'écriture. Si on en croit la notice, la caractéristique TEXTE de SOCRATE II permet de mémoriser $2^{15}-1$

octets, ce qui nous permet a priori n'importe quoi ! Mais quelques pages plus loin, on s'aperçoit que la récupération de l'information par un programme en langage évolué utilise une zone tampon de 140 caractères, et il faut donc découper les textes en tableaux de textes d'au plus 140 caractères.

Voulant clore ici, du moins provisoirement, la liste des avatars que nous avons subis dans la mise en oeuvre de SOCRATE [160], et voulant terminer sur une note optimiste, nous pensons qu'il serait intéressant de chercher dans la direction des types abstraits de données [51] une solution à ces différents problèmes : nous définirions des types de base (concept, étudiant, question) et des types construits à partir de ces types de base (leçons par exemple), ainsi que les différentes opérations que nous faisons dessus, sans nous occuper des problèmes de réalisation.

7.3. CARACTERISTIQUES DU PROGRAMME DE DIALOGUE.

Dans un souci d'uniformisation avec les autres logiciels du temps partagé du système SIRIS 8 (et en particulier l'éditeur de textes, dont l'utilisation par les étudiants est prévue), le programme de dialogue est une procédure cataloguée, commençant par le prologue, et mettant 5 commandes secondaires à la disposition de l'étudiant chaque fois qu'il reçoit le symbole '*', commandes correspondant au mode documentaire, au mode adaptation, au mode tutoriel, au mode récapitulatif et à l'épilogue.

Pendant chaque session, un fichier temporaire en (STS, MOD) permet de mémoriser un certain nombre de renseignements utiles au programme, par exemple le mode précédant le mode en cours, utilisé au moment de la transition d'un mode à l'autre (cf. 5.4.2). Le nom de ce fichier est un paramètre de la procédure cataloguée (% NOM).

Dans le schéma du § 7.1., nous avons indiqué une communication directe entre les modes documentaire et tutoriel et l'environnement extérieur à SATIRE : il s'agit, on le sait, d'aller utiliser les autres ressources du temps partagé pour répondre aux questions, lesdites questions étant des algorithmes ou des programmes à écrire.

Une procédure cataloguée ayant un seul point d'entrée, c'est au prologue qu'échoit la tâche de faire reprendre le travail là où il a été interrompu. Cela est possible grâce à la distinction entre deux commandes secondaires destinées à quitter la procédure :

* PAUSE correspondant à une interruption provisoire pour faire autre chose (avec nécessité de mettre le terminal dans l'état dormant grâce à la commande WAIT)

* FIN correspondant à l'interruption définitive (précédant directement le LOGOUT)

La commande PAUSE est mémorisée dans le fichier propre à la session. Le prologue reprend donc, dans ce cas, l'exécution du mode adaptation ou celui du mode tutoriel. La commande FIN détruit le fichier propre à la session.

ANNEXE 7A

Description théorique de la base

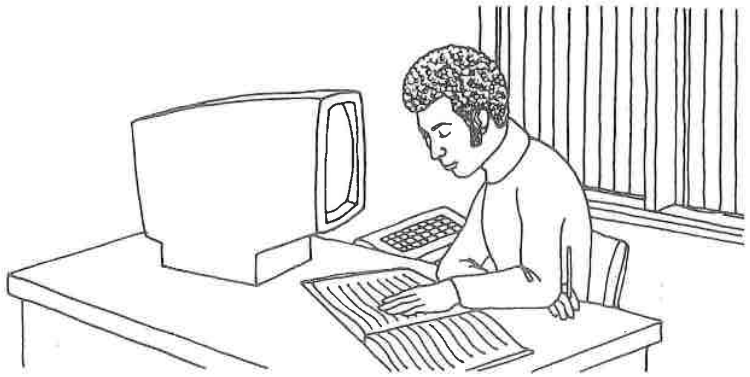
Syntaxe des caractéristiques	Sémantique des caractéristiques
ENTITE * concept	
DEBUT	
nom MOT *	
discipline MOT *	
TABLEAU (*) prérequis-de BINAIRE * DE 1 A *	les leçons dont le concept est prérequis
TABLEAU (*) relations	
DEBUT	
nom-relation NONCODEE (* *) (A,B,C,...)	relations utilisées pour la construction assistée de leçons
TABLEAU (*) antécédents BINAIRE * DE 1 A *	
FIN	
TABLEAU (*) étudié-dans	
DEBUT	
num-leçon BINAIRE * DE 1 A *	donne accès aux leçons dont le concept fait partie
nu BINAIRE * DE 1 A *	exemples de fonctions pédagogiques -cf. 3.3.2.4-
bèta BINAIRE * DE 1 A *	
I BINAIRE * DE 1 A *	
seuil BINAIRE * DE 1 A *	seuil critique d'erreurs au delà duquel l'étudiant a besoin d'une révision
FIN	l'accès à ces tableaux est un accès associatif
TABLEAU (*) étudié-par	
DEBUT	
num-étudiant BINAIRE * DE 1 A *	
erreurs-commises BINAIRE * DE 1 A *	renseignements nécessaires au processus de révision ; l'accès à ces tableaux est un accès associatif.
version-à-présenter NONCODEE (* 1) (R,1,2,...)	
FIN	
ENTITE * contexte	
DEBUT	
nom NONCODEE (* *) (-,=,...)	
public REFERE UN item	
TABLEAU * applications	
DEBUT	
num-leçon BINAIRE * DE 1 A *	questions qui seront posées à l'étudient après l'étude du concept (elles dépendent du contexte ce qui explique leur emplacement à ce niveau).
TABLEAU (*) questions BINAIRE * DE 1 A *	
FIN	
ENTITE * version	
DEBUT	
nom NONCODEE (* 1) (R, 1,2,...)	
contenu REFERE UN item PAR sujet-de	
TABLEAU (*) étudiée-par BINAIRE * DE 1 A *	les étudiants ayant "reçu" cette version de contexte de concept.
FIN	
ENTITE * intervention	
DEBUT	
version-concernée NONCODEE (* 1) (R,1,2,...)	l'intervention de l'étudiant concerne en principe la version qui lui a été présentée : nous avons fait ce choix d'implémentation pour économiser de la place, sachant que cela ne complique pas l'exploitation.
contenu TEXTE *	
FIN	
FIN	
FIN	
prérequis-demandé INVERSE TOUT concept	concepts que les étudiants ont souhaité trouver dans la base.

ENTITE * item	
DEBUT	
énoncé TEXTE *	
sujet-de REFERE UNE version DE UN contexte DE UN concept PAR contenu	(référence associée)
énoncé-de REFERE UNE question PAR énoncé	(référence associée)
solution-de REFERE UNE question PAR solution	(référence associée)
type NONCODEE (* 1) (A,B,C,...)	indique la nature de l'item
FIN	
ENTITE * leçon	
DEBUT	
nom MOT *	
total-concepts BINAIRE * DE 1 A *	nombre total de concepts de la leçon
temps-total FLOTTANT * DE 1 A *	temps moyen total à passer sur la leçon
TABLEAU (*) concepts-prérequis BINAIRE * DE 1 A *	les prérequis de la leçon
TABLEAU (*) contextes-possibles NONCODEE (* *) (-,=,-)	contextes dans lesquels la leçon est disponible
TABLEAU (*) fonction-d'ordre	
DEBUT	
num-concept BINAIRE * DE 1 A *	
type NONCODEE (2 1) (A,R)	précise si le concept est nouveau ou répété (cf. 3.4.1)
temps-moyen FLOTTANT * DE 1 A *	temps moyen à passer jusqu'à ce concept
FIN	liste des concepts de la leçon
étude-en-cours BINAIRE * DE 1 A *	nombre d'étudiants en train d'apprendre cette leçon
ENTITE * intervention	
DEBUT	
contenu TEXTE *	
FIN	
FIN	
ENTITE * question	
DEBUT	
énoncé REFERE UN item PAR énoncé-de	
solution REFERE UN item PAR solution-de	
TABLEAU (*) réponses-des-étudiants	
temps FLOTTANT * DE 1 A *	temps de réponse
valeur MOT *	contenu de la réponse (forme à définir selon leçon)
TABLEAU (*) concepts-composants BINAIRE * DE 1 A *	concepts dont la connaissance est supposée acquise
TABLEAU (*) est-dans-leçon BINAIRE * DE 1 A *	leçons auxquelles la question appartient
ENTITE * intervention	
DEBUT	
contenu TEXTE *	
FIN	
FIN	

```
ENTITE * étudiant
DEBUT
    nom MOT *
    prénom MOT *
    adresse
    DEBUT
        numéro MOT *
        voie MOT *
        commune MOT *
        code-postal MOT 5
        bureau-distributeur MOT *
    FIN
    contexte NONCODEE (* *) (-,-,-,...)          contexte habituel de l'étudiant
    ENTITE * apprentissage
    DEBUT
        concerne REFERE UNE leçon
        contexte NONCODEE (* *) (-,-,-,...)      contexte choisi pour la leçon dans les contextes disponibles
        version NONCODEE (* 1) (R,1,2,...)       version habituellement présentée
        nombre-de-questions BINAIRE * DE 1 A *   nombre de questions habituellement présentées
        concept-en-cours REFERE UN concept
        question-en-cours REFERE UNE question
        TABLEAU (*) révisions
        DEBUT
            concepts BINAIRE * DE 1 A *          pile qui gère les révisions imbriquées
            question BINAIRE * DE 1 A *
        FIN
        temps-passé FLOTTANT * DE 1 A *          temps passé par l'étudiant
        concepts-restants BINAIRE * DE 1 A *     concepts restant encore à acquérir
        TABLEAU (*) concepts-acquis BINAIRE * DE 1 A *   ne contient pas les concepts présentés par le mode adaptation
    FIN
FIN


ENTITE * comportement
DEBUT                                            un contenu est soit un changement de mode, soit un
    sujet REFERE UN étudiant                     changement de paramètre, soit un numéro de concept,
    TABLEAU (*) contenu MOT *                    soit un numéro de question, soit une indication
FIN                                              d'intervention


ENTITE * intervention
DEBUT
    contenu TEXTE *
FIN
```

CHAPITRE 8 : Programmation des différentes fonctions de SATIRE.

8.1. STRUCTURE DE LA BASE DE DONNEES FINALEMENT RETENUE.

Le description de structure ci-après correspond à celle de la base "modèle réduit" servant actuellement à la mise au point des programmes.

On notera que, dans l'impossibilité de traiter depuis COBOL des textes de plus de 140 caractères (cf. § 7.3.2.), les énoncés d'items sont devenus des chaînes de lignes.

```
 STRULT
BASE      :SATIRE-BASE
*F/:STRUCT/
00001*1,$L
00001 $BASE 50
00002 ENTITE 10 CONCEPT
00003 DEBUT
00004      NOM MOT 48
00005      DISCIPLINE MOT 20
00006 TABLEAU(8) ETUDIE-PAR
00007      DEBUT
00008          NUM-ETUDIANT BINAIRE 2 DE 1 A 8
00009          ERREURS-COMMISES BINAIRE 2 DE 1 A 100
00010          VERSION-A-PRESENTER NONCODEE (3 1) (R 1 2)
00011      FIN
00012     TABLEAU (5) PREREQUIS-DE BINAIRE 2 DE 1 A 5
00013     TABLEAU (10) ANTECEDENTS BINAIRE 2 DE 1 A 10
00014     TABLEAU (5) ETUDIE-DANS
00015          DEBUT
00016              NUM-LECON BINAIRE 2 DE 1 A 5
00017              SEUIL BINAIRE 2 DE 1 A 100
00018          FIN
00019      ENTITE 3 CONTEXTE
00020      DEBUT
00021          NOM NONCODEE (3 20) (LITTERAIRE SPECIALISTE SCIENTIFIQUE )
00022          TABLEAU (5) APPLICATIONS
00023                DEBUT
00024                    NUM-LECON BINAIRE 2 DE 1 A 5
00025                    TABLEAU (7) QUESTIONS BINAIRE 2 DE 1 A 60
00026                FIN
00027      PUBLIC REFERE UN ITEM
00028          ENTITE 3 VERSION
00029          DEBUT
00030              NOM NONCODEE(3 1) (R 1 2)
00031             TABLEAU(20) ETUDIEE-PAR BINAIRE 2 DE 1 A 20
00032             CONTENU REFERE UN ITEM PAR SUJET-DE
00033         FIN
00034          ENTITE 5 INTERVENTION
00035          DEBUT
00036              VERSION-CONCERNEE NONCODEE (3 1) (R 1 2)
00037               CONTENU TEXTE 140
00038          FIN
00039      FIN
00040 FIN
00041 PREREQUIS-DEMANDE INVERSE TOUT CONCEPT
```

```
00042 ENTITE 5 LECON
00043 DEBUT
00044      NOM MOT 20
00045      TOTAL-CONCEPTS BINAIRE 2 DE 1 A 10
00046      TEMPS-TOTAL FLOTTANT 4 DE 1 A 60
00047      TABLEAU (10) CONCEPTS-PREREQUIS BINAIRE 2 DE 1 A 10
00048      TABLEAU (3) CONTEXTES-POSSIBLES NONCODEE (3 20)
00049          (LITTERAIRE SPECIALISTE SCIENTIFIQUE )
00050      ETUDE-EN-COURS BINAIRE 2 DE 1 A 20
00051      TABLEAU(20) LEC-ETUDIE-PAR BINAIRE 2 DE 1 A 20
00052      TABLEAU(6) FONCTION-DORDRE
00053         DEBUT
00054             NUM-CONCEPT BINAIRE 2 DE 1 A   10
00055             TEMPS-MOYEN  FLOTTANT 4 DE 1 A 60
00056             TYPE NONCODEE (2 1) (A R)
00057         FIN
00058        ENTITE 5 INTERVENTION
00059        DEBUT
00060            CONTENU TEXTE 140
00061        FIN
00062 TITRE REFERE UN ITEM PAR TITRE-LEC
00063 FIN
```

```
00080 ENTITE 20 ETUDIANT
00081 DEBUT
00082      NOM MOT 20
00083      PRENOM MOT 15
00084      ADRESSE
00085         DEBUT
00086         NUMERO MOT 4
00087         VOIE MOT 30
00088         COMMUNE MOT 20
00089         CODE-POSTAL MOT 5
00090         BUREAU-DISTRIBUTEUR MOT 20
00091         FIN
00092      CONTEXTE NONCODEE(3 20) (LITTERAIRE SPECIALISTE SCIENTIFIQUE )
00093      ENTITE 3 APPRENTISSAGE
00094      DEBUT
00095          VERSION NONCODEE(3 1) (R 1 2)
00096          CONTEXTE NONCODEE (3 20) (LITTERAIRE SPECIALISTE SCIENTIFIQUE )
00097          NOMBRE-DE-QUESTIONS BINAIRE 2 DE 1 A 10
00098          CONCEPT-EN-COURS BINAIRE 2 DE 1 A 6
00099          QUESTION-EN-COURS BINAIRE 2 DE 1 A 5
00100          TABLEAU(10) CONCEPT-ACQUIS BINAIRE 2 DE 1 A 10
00101          TABLEAU(10) REVISIONS
00102             DEBUT
00103             CONCEPT-A BINAIRE 2 DE 1 A 10
00104             QUESTION-A BINAIRE 2 DE 1 A 5
00105             FIN
00106          TEMPS-PASSE FLOTTANT 4 DE 1 A 60
00107          CONCEPTS-RESTANTS BINAIRE 2 DE 1 A 10
00108          CONCERNE REFERE UNE LECON
00109       FIN
00110 FIN
```



```
00064 ENTITE 60 QUESTION
00065 DEBUT
00066      TABLEAU (15) REPONSES-ETUDIANTS
00067          DEBUT
00068              TEMPS FLOTTANT 4 DE 1 A 15
00069              VALEUR MOT 20
00070          FIN
00071      TABLEAU(10) CONCEPT-COMPOSANT BINAIRE 2 DE 1 A 10
00072      TABLEAU(5) EST-DANS-LECON BINAIRE 2 DE 1 A 5
00073      ENONCE REFERE UN ITEM PAR ENONCE-DE
00074      SOLUTION REFERE UN ITEM PAR SOLUTION-DE
00075        ENTITE 5 INTERVENTION
00076        DEBUT
00077            CONTENU TEXTE 140
00078        FIN
00079 FIN
```

```
00111 ENTITE 100 ITEM
00112 DEBUT
00113      TYPE NONCODEE(3 1) (A B C)
00114       SUJET-DE REFERE UNE VERSION DE UN CONTEXTE DE UN CONCEPT PAR
00115           CONTENU
00116      ENONCE-DE REFERE UNE QUESTION PAR ENONCE
00117      SOLUTION-DE REFERE UNE QUESTION PAR SOLUTION
00118      TITRE-LEC REFERE UNE LECON PAR TITRE
00119      CH-LIG CHAINE 4 TOUTE LIGNE PAR REF-IT
00120 FIN
00121 ENTITE 10 COMPORTEMENT
00122 DEBUT
00123      TABLEAU (10) CONTENU MOT 40
00124      SUJET REFERE UN ETUDIANT
00125 FIN
00126 ENTITE 2000 LIGNE
00127 DEBUT
00128      ENONCE1 TEXTE 140
00129  REF-IT REFERE UN ITEM PAR CH-LIG
00130 FIN
00131 ENTITE 20 COMMANDE
00132 DEBUT
00133      SYMBOLE MOT 3
00134      VALEURHEX MOT 8
00135 FIN
00136 *BASE
FIN DE FICHIER
```

8.2. FONCTIONS PROGRAMMEES DANS LE LANGAGE DE REQUÊTES SOCRATE II.

Nous donnons ici la structure et un exemple d'exécution (toujours sur la base modèle réduit) de quelques fonctions programmées dans le langage de requête SOCRATE.

```
 PRECOMP
*F/LISTE-DES-ETUDIANTS/
00001*1,$L
00001 I NOM DE TOUT ETUDIANT
FIN DE FICHIER
*
```

```
 GO LISTE-DES-ETUDIANTS
NOM : PISTER
NOM : SCHUSTER
NOM : SCHUSTER
NOM : BERTAUD
$
```

```
 F/PREREQUIS-DEMANDES/
00001*1,$L
00001 M X1 = UN PREREQUIS-DEMANDE
00002 FAIRE
00003 SI SYS-ANOM = X'01D7' ALORS SORTIE
00004                 SINON
00005                 I NOM DE X1
00006                 I DISCIPLINE DE X1
00007                 M X1 = SUIVANT DE X1
00008                 REFAIRE
00009        FIN
00010 FIN
FIN DE FICHIER
*
```

```
 GO PREREQUIS-DEMANDES
NOM : NOMBRES-ENTIERS
DISCIPLINE : ALGORITHMIQUE
NOM : LES EXPRESSIONS, LES FONCTIONS STANDARDS
DISCIPLINE : ALGORITHMIQUE
* 7     FIN DE CHAINE
$
```

```
 F/CONTEXTES-CONCEPT/
00001*1,$L
00001 M Z2 = 'NUMERO DU CONCEPT :'
00002 ECRIRE Z2
00003 LIRE DANS Z1
00004 M Y1 = Z1
00005 I NOM DE TOUT CONTEXTE DE UN CONCEPT Y1
FIN DE FICHIER
*
```



```
 GO CONTEXTES-CONCEPT
NUMERO DU CONCEPT :
?1
NOM : SPECIALISTE
$GO CONTEXTES-CONCEPT
NUMERO DU CONCEPT :
?3
NOM : SPECIALISTE
$
```

```
 F/ETUDIANT-CONCEPT/
00001*1,$L
00001 M Z1 = 'NUMERO DU CONCEPT'
00002 ECRIRE Z1
00003 LIRE DANS Z2
00004 M Y1 = Z2
00005 M Y2 = 0
00006 M Y4 = NBDE TOUT ETUDIANT
00007 FAIRE
00008     M Y2 = Y2 + 1
00009     SI Y2 > Y4 ALORS SORTIE
00010                SINON
00011 SI NUM-ETUDIANT DE ETUDIE-PAR (Y2) DE UN CONCEPT Y1
00012          = U ALORS SORTIE
00013                          SINON
00014                M Y3 = NUM-ETUDIANT DE ETUDIE-PAR (Y2)
00015          DE UN CONCEPT Y1
00016                I NOM DE UN ETUDIANT Y3
00017                REFAIRE
00018                FIN
00019        FIN
00020 FIN
FIN DE FICHIER
*
```

```
 GO ETUDIANT-CONCEPT
NUMERO DU CONCEPT
?3
NOM : SCHUSTER
$GO ETUDIANT-CONCEPT
NUMERO DU CONCEPT
?1
NOM : SCHUSTER
$GO ETUDIANT-CONCEPT
NUMERO DU CONCEPT
?2
$
```

```
 F/ETAT/
00001*1,$L
00001 M Z1 = 'NOMBRE DE CONCEPTS:'
00002  M Z2 = NBDE TOUT CONCEPT
00003 ECRIRE Z1 ECRIRE Z2
00004 M Z1 = 'NOMBRE DE LECONS'
00005 M Z2 = NBDE TOUTE LECON
00006 ECRIRE Z1 ECRIRE Z2
00007 M Z1 = 'NOMBRE DE QUESTIONS'
00008 M Z2 = NBDE TOUTE QUESTION
00009 ECRIRE Z1 ECRIRE Z2
00010 M Z2 = NBDE TOUT ETUDIANT
00011 M Z1 = 'NOMBRE D ETUDIANTS'
00012 ECRIRE Z1 ECRIRE Z2
00013 M Z2 = NBDE TOUT ITEM
00014 M Z1 = 'NOMBRE D ITEMS'
00015 ECRIRE Z1 ECRIRE Z2
FIN DE FICHIER
*


 GO ETAT
NOMBRE DE CONCEPTS:
4
NOMBRE DE LECONS
5
NOMBRE DE QUESTIONS
0
NOMBRE D ETUDIANTS
4
NOMBRE D ITEMS
0
$
```

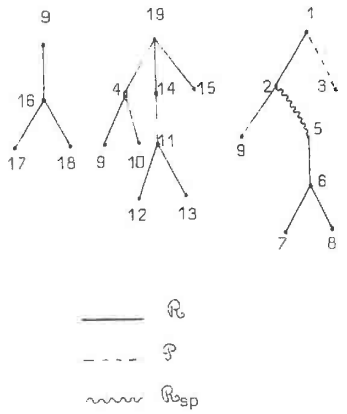
8.3. CONSTRUCTION ASSISTEE DE LECONS.

Nous avions primitivement pensé écrire des programmes conversationnels de construction assistée de leçons : le professeur, interrogé par le programme, aurait donné la liste des concepts destinés à la leçon prévue (avec, à la condition qu'ils ne figurent pas déjà dans la base, introduction de ces concepts) ; il aurait également fourni des relations entre ces concepts. Un autre programme aurait alors calculé l'analyse de contenu. C'est ensuite au moment de la programmation du contenu que, de manière à nouveau conversationnelle, le professeur aurait été invité à donner les compléments de la leçon : questions associées aux concepts, attributs divers du type de ceux présentés en 3.3.2.4.

Vu la relative complexité des algorithmes du chapitre 3, nous avions prévu d'écrire les programmes en PL/1, COBOL étant peu adapté.

Les nombreuses difficultés rencontrées lors de la programmation du dialogue étudiant-ordinateur, et des informations reçues selon lesquelles l'utilisation de SOCRATE depuis PL/1 provoquait des conflits de gestion de mémoire, nous ont fait abandonner ce projet. Nous avons donc séparé la construction assistée de leçons en deux parties distinctes. La seconde, qui consiste à introduire la leçon construite, sera présentée au paragraphe suivant. C'est la première, à savoir les programmes d'analyse et de programmation du contenu devenus maintenant indépendants de la base, dont nous donnons des exemples ici. Ils ont effectivement été programmés en PL/1.

Nous avons utilisé pour représenter les ramifications une représentation classique avec lien horizontal et lien vertical [104,140,147,210] augmentée des différents attributs utilisés par les algorithmes (nature des liens, taille). La figure 21 donne un exemple de ramification avec sa représentation. Le tableau "natlh" indique la nature du lien qui unit le frère au père commun.



		natlv				"natlh"			
	tal	lv	$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$	lh	$\mathcal{R}$	$\mathcal{R}_{sp}$	$\mathcal{P}$	t
1	9	2	1	0	0	5	0	0	0	4
2	16	3	1	0	0	0	0	0	0	3
3	17	0	0	0	0	4	1	0	0	1
4	18	0	0	0	0	0	0	0	0	1
5	19	6	1	0	0	14	0	0	0	9
6	4	9	1	0	0	7	1	0	0	3
7	14	11	1	0	0	8	1	0	0	4
8	15	0	0	0	0	0	0	0	0	1
9	9	0	0	0	0	10	1	0	0	1
10	10	0	0	0	0	0	0	0	0	1
11	11	12	1	0	0	0	0	0	0	3
12	12	0	0	0	0	13	0	0	0	1
13	13	0	0	0	0	0	0	0	0	1
14	1	2	1	0	0	0	0	0	0	8
15	2	17	1	0	0	16	0	0	1	6
16	3	0	0	0	0	0	0	0	0	1
17	9	0	0	0	0	18	0	1	0	1
18	5	19	1	0	0	0	0	0	0	4
19	6	20	1	0	0	0	0	0	0	3
20	7	0	0	0	0	21	1	0	0	1
21	8	0	0	0	0	0	0	0	0	1

figure 21 : Exemple de ramification avec une représentation

Nous rappelons que les algorithmes sont spécifiques des relations qui existent entre les concepts : les algorithmes que nous programmons actuellement correspondent aux relations $\mathcal{R}$, $\mathcal{R}_{sp}$ et $\mathcal{P}$ des leçons d'algorithmique et de programmation du chapitre 6.

8.3.1. *Premier algorithme d'analyse du contenu (relations $\mathcal{R}$ et $\mathcal{R}_{sp}$).*

L'algorithme présenté en 3.3.2.3, transformé en 3.4.2, est modifié ainsi :

- dans la pile : tous les points qui sortent de la pile par épuisement de leurs prédécesseurs sont condamnés ; dans le cas $s(x) > 1$, la seule action possible est : i est une sortie de x ;
- remplacer partout $\mathcal{R}$ par $\mathcal{R}$ ou $\mathcal{R}_{sp}$ et $\mathcal{R}$ ou $\mathcal{P}$ par $\mathcal{R}$ ou $\mathcal{R}_{sp}$.

On trouvera ci-après la description de l'algorithme.

8.3.2. *Premier algorithme de programmation de contenu.*

Dans cet algorithme, on a fait jouer à $\mathcal{R}$ et à $\mathcal{R}_{sp}$ le même rôle.

Nous utilisons deux procédures récursives LEC et DER ayant pour paramètres la ramification à lire, caractérisée par l'indice de sa racine la plus à gauche, et la dernière étiquette lue. Les valeurs des étiquettes sont converties en chaînes de caractères, ce qui permet d'utiliser la concaténation pour décrire le résultat de LEC qui appartient à V^* (cf. 3.4.3.).

Pour réaliser la "séparation" des arborescences d'une ramification, on utilise un artifice qui consiste à détruire le lien horizontal.

L'algorithme de reconnaissance est alors le suivant :

- si l'indice est nul : il s'agit de la ramification vide.
- si l'indice est non nul et que le lien horizontal est différent de -1 : c'est une somme ; on retourne alors, outre l'indice, l'indice du frère, et le lien horizontal est mis à -1.
- si l'indice est non nul et que le lien horizontal est égal à -1 : c'est une arborescence ;
  - si le lien vertical est nul : c'est une feuille
  - si le lien vertical est non nul : il faut alors retourner les indices de tous les fils en utilisant les liens horizontaux successifs, trier par tailles décroissantes et changer en conséquence les liens horizontaux.

- rami : ramification représentée par les tableaux val(ent), ℓv(ent), $n\ell v1$(bool), $n\ell v2$(bool), ℓh(ent), $n\ell h1$(bool), $n\ell h2$(bool), t(ent) (figure 21).
- N(ent) : dimension du graphe
- g(bool tab N) : tableau qui indique les sommets qui appartiennent à g
- prérequis(bool tab N) : tableau qui indique les prérequis dans G
- cond(bool tab N+1) : tableau qui indique les sommets condamnés de G
- s(ent tab N+1) : fonction s de l'algorithme informel
- h(ent) : hauteur de la pile
- proch(ent tab N+2) : tableau qui indique, pour chaque sommet, le prochain prédécesseur à entrer dans la pile
- finpred(bool) : indique que le sommet de la pile n'a plus de prédécesseurs non encore entrés
- deb(ent tab N+2) } dictionnaire
- pred(ent tab P) } du graphe
- nbpuits(ent) nombre de sommets de demi-degré extérieur nul avant adjonction éventuelle de ω_0
- fin(ent) premier emplacement libre de pred avant adjonction éventuelle de ω_0
- ℓ(ent) indice du dernier emplacement occupé de rami
- α(ent) indice dans rami du frère du sommet qu'on empile
- npr1, npr2(bool tab P) nature des relations dans le graphe
- tp(ent tab N + 1) taille provisoire des branches
- puits(ent tab N) mémorise les sommets de demi-degré extérieur nul avant adjonction de ω_0
- ind(ent) indice dans pred du sommet à empiler
- p(ent tab N + 1) pile de l'algorithme informel
- mem(ent tab N + 1) mémorise l'ancienne valeur de s avant sa modification

```
    module principal
3   rami : Iniarbo ; jqa finarbo rep arbo
1   N = donnée
2   prerequis, deb, pred, fin, npr1, npr2, dde = donnés

    arbo (construit une arborescence de la ramification)
1   rami, cond : inipile ; jqa finpile rep pile                avec s_f
6   finarbo : finarbo = vrai ; pour I dans 1 → N jqa Finarbo = faux rep
                        si g(I) alors finarbo = finarbo et prérequis(I)
3   g, s_1, mem, dde : pour I dans 1 → N1 rep { mem(I) = s̄(I)
                                              { si cond(I) et G(I) alors misajour
4   s_2, nbpuits : nbpuits = 0 ; pour I dans 1 → N rep si dde(I) = 0 alors { nbpuits = nbpuits + 1
                                                                           { S(I) = 0
                                                                           { puits(nbpuits) = I
                                                  sinon S(I) = mem(I)
5   s_f, pred, dde_f : si nbpuits ≠ 1 alors adjonction avec s_2
2   N1 : si nbpuits ≠ 1 alors N + 1 sinon N

    pile (voir algorithme informel du § 3.3.2.3)
2   rami,h : si s(p(h̄)) > 1 alors JÔpile1 sinon si finpred alors depile2 sinon empile
             avec ind
1   finpred,ind : si proch(p(h)) = deb(p(h)+1) alors finpred = vrai
                                                 sinon { finpred = faux
                                                       { ind = proch(p(h))
                                                       { proch(p(h)) = proch(p(h)) + 1
3   finpile = (h = 0)

    misajour (construction du nouveau graphe par suppression des points condamnés)
1   g(I) = faux
2   s,nbpuits : pour J dans deb(I) → deb(I+1)-1 rep { s(pred(J)) = s̄(pred(J)) - 1
                                                    { si s(pred(J)) = 0
                                                        alors { nbpuits = nbpuits + 1
                                                              { puits(nbpuits) = pred(J)

    adjonction (de ω_0)
1   s(n + 1) = 0
2   dde(n + 1) = 0
3   s, pred : pour J dans 1 → nbpuits rep { pred(deb(N + 1) + J - 1) = puits(J)
                                          { npr1(deb(N + 1) + J - 1) = vrai
                                          { npr2(deb(N + 1) + J - 1) = faux
                                          { s(puits(J)) = 1
                                          { dde(puits(J)) = 1
4   deb(N + 2) = fin + nbpuits

    empile
1   h = h̄ + 1
2   p(h) = pred(ind)
3   ℓ = ℓ̄ + 1
4   val(ℓ) = pred(ind)
5   ℓv, ℓh, nℓv1, nℓv2, nℓh1, nℓh2 :
      si ind = deb(p(h̄)) alors ℓv(ℓ̄) = ℓ ; nℓv1(ℓ̄) = npr1(ind) ; nℓv2(ℓ̄) = npr2(ind)
                         sinon ℓh(α) = ℓ ; nℓh1(α) = npr1(ind) ; nℓh2(α) = npr2(ind)
6   tp : pour K dans 1 → h rep tp(p(K)) = tp̄(p(K)) + 1
```

```
dépile 1   (on dépile un sommet dès qu'il a plus d'un successeur mais il
            n'est pas condamné)
8  h = h̄ - 1
1  frère = p(h̄)
2  proch(p(h̄)) = deb(p(h̄))
4  t(α) = tp(p(h̄))
3  α : α = ℓ ; jqa val(α) = frère rep α = ᾱ - 1
5  ℓv(α) = 0

dépile 2   (on dépile parce qu'on a essayé tous les prédécesseurs ; le sommet
            est condamné)
7  h = h̄ - 1
1  frère = p(h̄)
2  proch(p(h̄)) = deb(p(h̄))
4  t(α) = tp(p(h̄))
3  α : α = ℓ ; jqa val(α) = frère rep α = ᾱ - 1
5  ℓv(α) = 0
6  cond(p(h̄)) = vrai

inipile   (on empile le sommet unique de demi-degré extérieur nul)
1  h = 1
2  p(h) = si nbpuits = 1 alors puits(1) sinon N + 1
9  tp(p(h)) = 1
3  finpile = faux
4  tp : pour I dans 1 → N + 1 rep tp(I) = 0
5  proch = deb
6  ℓ = ℓ̄ + 1
7  val(ℓ) = p(h)
8  si α ≠ -1 alors ℓh(ℓ) = α sinon ℓh(ℓ) = 0

Iniarbo

4  α = -1
8  finarbo : finarbo = vrai ; pour I dans 1 → N jusqua finarbo = faux rep
                    si g(I) alors finarbo = finarbo et prérequis(I)
1  g : pour I dans 1 → N rep g(I) = vrai
11 g(N + 1) = faux
2  cond : pour I dans 1 → N rep cond(I) = faux
3  ℓ = 0
7  s, pred, dde : si nbpuits ≠ 1 alors adjonction        avec s, puits
6  puits, nbpuits : nbpuits = 0 ; pour I dans 1 → N rep si s(I) = 0
                                                  alors { nbpuits = nbpuits + 1
                                                        { puits(nbpuits) = I
5  s = dde.
9  ℓv, ℓh = 0
10 nℓv1, nℓv2, nℓh1, nℓh2 = faux
```

- frère(ent) frère du sommet qu'on empile

- dde(ent tab N + 1) tableau des demi-degrés extérieurs des points du graphe

8.3.3. *Programme correspondant.*

```
 F/CONTANAL/
00001.000*1,$P
!LIMIT (TIME,3),(CORE,100),(SPDISC,100)
!PL2 SI,LS,GO,XREF,NEST,STMT
 CONTANAL:PROC OPTIONS(MAIN);
DCL VAL(1:20) DEC FIXED(2) INIT((20)0),FINARBO BIT(1) INIT('1'B),
    LV (1:20) DEC FIXED(2) INIT((20)0),FINPRED BIT(1),
    NLV1(1:20) BIT(1) INIT((20)'0'B),
    NLV2(1:20) BIT(1) INIT((20)'0'B),
OPERATOR INTERRUPT
00008.000*1,$P
!LIMIT (TIME,3),(CORE,100),(SPDISC,100)
!PL2 SI,LS,GO,XREF,NEST,STMT
 CONTANAL:PROC OPTIONS(MAIN);
DCL VAL(1:20) DEC FIXED(2) INIT((20)0),FINARBO BIT(1) INIT('1'B),
    LV (1:20) DEC FIXED(2) INIT((20)0),FINPRED BIT(1),
    NLV1(1:20) BIT(1) INIT((20)'0'B),
    NLV2(1:20) BIT(1) INIT((20)'0'B),
    LH (1:20) DEC FIXED(2) INIT((20)0),
    NLH1(1:20) BIT(1) INIT((20)'0'B),
    NLH2(1:20) BIT(1) INIT((20)'0'B),
    T  (1:20) DEC FIXED(2) INIT((20)0);
DCL (N,NBPUITS,FIN,IND,H,ALFA,FRERE) DEC FIXED(2) ;
DCL L DEC FIXED(2) INIT(0);
DCL PRED(1:15) DEC FIXED(2),
    NPR1(1:15) BIT(1),
    NPR2(1:15) BIT(1);
GET LIST(N);
  BEGIN;
  DCL COND(1:N+1) BIT(1) INIT ((1000)'0'B),
      PUITS(1:N) DEC FIXED(2) INIT((1000)0),
      S(1:N+1) DEC FIXED (2),
      DDE(1:N+1) DEC FIXED(2),
      MEM(1:N+1) DEC FIXED(2),
      PREREQUIS(1:N) BIT(1),
      DEB(1:N+2) DEC FIXED(2),
      G(1:N+1) BIT(1),
      PROCH(1:N+2) DEC FIXED(2),
      P(1:N+1) DEC FIXED(2) INIT((1000)0),
      TP(1:N+1) DEC FIXED(2);
  GET LIST (DDE,FIN,PREREQUIS,NPR1,NPR2,PRED);   GET LIST (DEB);
PUT LIST (FINARBO,FINPILE,FINPRED,L,N,FIN);PUT SKIP;
PUT LIST (DDE);PUT SKIP;
PUT LIST (PREREQUIS);PUT SKIP;
PUT LIST (NPR1);PUT SKIP;
PUT LIST (NPR2);PUT SKIP;
PUT LIST (COND);PUT SKIP;
PUT LIST (DEB);PUT SKIP;
PUT LIST (G);PUT SKIP;
PUT LIST (PRED);PUT SKIP;
```

```
  /*INIARBO*/
  DO J=1 TO N+1;G(J)='1'B;S(J)=DDE(J);END;G(N+1)='0'B;
  NBPUITS=0;DO I=1 TO N;
            IF S(I)=0 THEN DO;NBPUITS=NBPUITS+1;PUITS(NBPUITS)=I;END;
             END;
DO M=1 TO NBPUITS;PUT LIST(PUITS(M));END;PUT SKIP;
  IF NBPUITS > 1 THEN DO; /*ADJONCTION*/
                        S(N+1),DDE(N+1)=0;
                        G(N+1)='1'B;
                        DO J=1 TO NBPUITS;
                        PRED(DEB(N+1)+J-1)=PUITS(J);
                        NPR1(DEB(N+1)+J-1)='1'B;NPR2(DEB(N+1)+J-1)='0'B;
                        S(PUITS(J)),DDE(PUITS(J))=1;
                        END;
                        DEB(N+2)=FIN+NBPUITS;
                         END; /*ADJONCTION*/
  ALFA=-1;
  I=1;DO WHILE(FINARBO='1'B & I<N+1);
      IF G(I)='1'B THEN FINARBO=FINARBO & PREREQUIS(I);
       I=I+1;
       END;
  /*FININIARBO*/
  DO WHILE(FINARBO='0'B);
    /*ARBO*/
    /*INIPILE*/H=1;DO I=1 TO N+1;TP(I)=0;END;PROCH=DEB;
                IF NBPUITS=1 THEN P(H)=PUITS(1);ELSE P(H)=N+1;TP(P(H))=1;
DO M=1 TO H;PUT LIST (P(M));END;PUT SKIP;
                L=L+1;VAL(L)=P(H);
                IF ALFA^=-1 THEN LH(L)=ALFA; ELSE LH(L)=0;
    DO WHILE(H>0);
      /*PILE*/
       IF PROCH(P(H))=DEB(P(H)+1) THEN FINPRED='1'B;
                                  ELSE DO;
                                       FINPRED='0'B;
                                       IND=PROCH(P(H));
                                       PROCH(P(H))=PROCH(P(H))+1;
                                       END;
      IF S(P(H))>1 THEN DO;/*DEPILE1*/
                        FRERE=P(H);
                        PROCH(P(H))=DEB(P(H));
                        ALFA=L;DO WHILE (VAL(ALFA)^=FRERE);ALFA=ALFA-1;END;
                        T(ALFA)=TP(P(H));
                        H=H-1;
DO M=1 TO H;PUT LIST(P(M));END;PUT SKIP;
                        END;/*DEPILE1*/
                    ELSE IF (FINPRED='1'B) THEN DO;/*DEPILE2*/
                                                FRERE=P(H);
                                                PROCH(P(H))=DEB(P(H));
                                                ALFA=L;
                                                DO WHILE (VAL(ALFA)^=FRERE);
                                                   ALFA=ALFA-1;
                                                END;
                                                T(ALFA)=TP(P(H));
                                                COND(P(H))='1'B;
                                                H=H-1;
DO M=1 TO H;PUT LIST(P(M));END;PUT SKIP;
                                                END;/*DEPILE2*/
```



```
                    ELSE DO;/*EMPILE*/
                         H=H+1;L=L+1;
                         P(H),VAL(L)=PRED(IND);
                         IF IND=DEB(P(H-1))THEN DO;
                                                LV(L-1)=L;
                                                NLV1(L-1)=NPR1(IND);
                                                NLV2(L-1)=NPR2(IND);
                                                END;
                                            ELSE DO;
                                                LH(ALFA)=L;
                                                NLH1(ALFA)=NPR1(IND);
                                                NLH2(ALFA)=NPR2(IND);
                                                END;
                         DO K=1 TO H  ; TP(P(K))=TP(P(K))+1;END;
DO M=1 TO H;PUT LIST(P(M));END;PUT SKIP;
                         END;/*EMPILE*/
    END;/*PILE*/
    IF NBPUITS>1 THEN N1=N+1 ;ELSE N1=N;
    DO I=1 TO N1;
        MEM(I)=S(I);
            IF COND(I)='1'B & G(I)='1'B THEN DO;/*MISAJOUR*/
                              G(I)= '0'B;
                              DO J=DEB(I) TO DEB(I+1)-1;
                               S(PRED(J))=S(PRED(J))-1;
                               DDE(PRED(J))=DDE(PRED(J))-1;
                              END;
                              END;/*MISAJOUR*/
            END;
  NBPUITS=0;DO I=1 TO N;
              IF G(I) THEN DO;
                    IF DDE(I)=0 THEN DO;
                                NBPUITS=NBPUITS+1;
                                 S(I)=0;
                                 PUITS(NBPUITS)=I;END;
                               ELSE S(I)=MEM(I);
                             END;
            END;
    IF NBPUITS>1 THEN DO;/*ADJONCTION*/
                      S(N+1),DDE(N+1)=0;
                      DO J=1 TO NBPUITS;
                         PRED(DEB(N+1)+J-1)=PUITS(J);
                         NPR1(DEB(N+1)+J-1)='1'B;
                         NPR2(DEB(N+1)+J-1)='0'B;
                         S(PUITS(J)),DDE(PUITS(J))=1;
                       END;
                       DEB(N+2)=FIN+NBPUITS;
                        END;/*ADJONCTION*/
    FINARBO='1'B;I=1;
    DO WHILE(FINARBO='1'B & I<N+1 );
       IF G(I)='1'B THEN FINARBO=FINARBO & PREREQUIS(I);
       I=I+1;
    END;
  END;/*ARBO*/
DO M=1 TO L;PUT SKIP;
PUT LIST(VAL(M),LV(M),NLV1(M),NLV2(M),LH(M),NLH1(M),NLH2(M),T(M));
END;
  END;
END CONTANAL;
```

8.4. INTRODUCTION DE LECONS.

On a vu au paragraphe précédent que la fonction "construction assistée de leçons" et la fonction "introduction de leçons dans la base" avaient été dissociées. Notons que cela permet à l'enseignant d'introduire dans la base des leçons qu'il aurait construites selon ses propres méthodes.

Il suffit alors de se reporter à la partie de la structure de la base (§ 8.1.) qui concerne l'entité "leçon" pour faire apparaître les différentes caractéristiques auxquelles l'enseignant devra donner une valeur.

Cette fonction n'est actuellement pas encore opérationnelle. Elle est réalisée par un programme COBOL (exécuté de manière conversationnelle) qui permet d'introduire les items correspondants aux concepts et aux questions au cours du programme ou bien, l'ayant fait auparavant grâce à des requêtes SOCRATE (§ 8.2.), de les repérer par leur numéro de réalisation.

8.5. DIALOGUE ETUDIANT-ORDINATEUR.

Nous donnons ici, à titre d'exemple, la version (actuellement opérationnelle) du prologue et du mode documentaire, comprenant le listing du programme source, le listing des sous-programmes SOCRATE qu'il utilise, et un listing d'exécution.

8.5.1. *Extrait du programme source.*

```
!LIMIT (CORE,80),(SPDISC,30),(TIME,3)
$ASSIGN BO,MTN,FIL
!COBOL SI,LS,BO,OQ
        IDENTIFICATION DIVISION.
        PROGRAM-ID. KIC2.
        ENVIRONMENT DIVISION.
        CONFIGURATION SECTION.
        SOURCE-COMPUTER. CII-10070.
        OBJECT-COMPUTER. CII-10070.
        INPUT-OUTPUT SECTION.
        FILE-CONTROL.
            SELECT F-IMPR ASSIGN TO ECRI.
            SELECT F-LECT ASSIGN TO LIRE.
             SELECT TOTO ASSIGN TO ATOT,ACCESS MODE IS RANDOM
       -     , ACTUAL KEY IS CLET.
        DATA DIVISION.
```



```
FILE SECTION.
FD F-IMPR LABEL RECORD OMITTED.
01 LIGNE.
    02 L11 PIC X(140).
FD TOTO LABEL RECORD IS STANDARD DATA RECORD ARTICLE1.
01 ARTICLE1.
    02 CLET PIC 9.
    02 NOM-T PIC X(20).
    02 PRENOM-T  PIC X(15).
    02 NUM-REA-ETUD COMP PIC 99.
    02 MODE-EN-COURS PIC X.
    02 NUM-REA-APPR COMP  PIC  99.
    02 CONC-T  PIC  99.
    02 CONT-T  PIC  99.
    02 QUES-T  PIC  99.
FD F-LECT LABEL RECORD OMITTED.
01 REP1.
    02 REP  PIC X(70).
WORKING-STORAGE SECTION.
77 MDS PIC X.
77 TYP PIC X.
77 FIN-FICH COMP VALUE 471.
77 NCD COMP.
77 LCD COMP.
77 NSI COMP.
77 NBR COMP.
77 LCP COMP.
77 NCP COMP.
77 MDR PIC X.
77 SCR PIC X.
77 COR COMP.
77 UN COMP VALUE 1.
77 DEUX COMP VALUE 2.
77 QUAT COMP VALUE 4.
77 MOD-PREC PIC X.
77 LONG1 COMP VALUE 540.
77 NOM-ETUD PIC X(20).
77 PRENOM-ETUD PIC X(15).
77 CONTEXTE-ETUD PIC X(20).
77 EXISTE2 PIC X.
77 LONG2 COMP VALUE 138.
77 LONG3 COMP VALUE 534.
77 PAUSE PIC X.
77 I COMP.
77 COM-SEC PIC XXX.
77 ITEM-POSS PIC X(30) VALUE "ITEM-POSS".
77 RECH-ETUD PIC X(30) VALUE "RECH-ETUD".
77 RECH-ETUDS PIC X(30) VALUE "RECH-ETUDS".
77 CREAT-ETUD PIC X(30) VALUE "CREAT-ETUD".
77 VAR-X1 COMP VALUE 1.
77 INI-ETUD PIC X(30) VALUE "INI-ETUD".
77 LONG7 COMP VALUE 4.
77 FIN-BASE COMP VALUE 471.
77 VAR-X2 COMP VALUE 2.
77 PRES-ITEM PIC X(30) VALUE "PRES-ITEM".
77 LEC-CONC PIC X(30) VALUE "LEC-CONC".
77 PRES-CONT PIC X(30) VALUE "PRES-CONT".
77 NBRE-CONT PIC X(30) VALUE "NBRE-CONT".
77 RECH-CONT   PIC X(30) VALUE "RECH-CONT".
77 RECH-VERS   PIC X(30) VALUE "RECH-VERS".
77 MAJ-PRER     PIC X(30) VALUE "MAJ-PRER".
77 RECH-CONC    PIC X(30) VALUE "RECH-CONC".
```

```
 77 LONG4 COMP VALUE 8.
 77 LONG5 COMP VALUE 24.
 77 LONG6 COMP VALUE 31.
 77 LONG8 COMP VALUE 540.
 77 LONG9 COMP VALUE 28.
 77 LONG10 COMP VALUE 20.
 77 VAR-X3 COMP VALUE 3.
 77 VAR-X4 COMP VALUE 4.
 77 NOM-CONT PIC X(20).
 77 TROIS COMP VALUE 3.
 77 NUM-CONC PIC 99.
 77 ICONT PIC 9.
 77 ICONC PIC 99.
 01 TABLE-INTERFACE.
     02 NOM-BASE        PIC X(12) VALUE "SATIRE-BASE".
     02 NOM-S-STRUCT    PIC X(12) VALUE ":STRUCT".
     02 NOM-UT          PIC X(12) VALUE "KEMIAA19".
     02 NUM-COMPTE      PIC X(4)  VALUE "MA19".
     02 M-D-P           PIC X(8)  VALUE "SATIRE".
     02 ANOMALIE.
        03 CODE-ANOMALIE COMP.
        03 LG-SS-PG     PIC X.
        03 NOM-SS-PG    PIC X(30).
        03 FILLER       PIC X.
     02 FILLER    PIC X(2636).
*            ***  DESCRIPTIONS DE  L"ESPACE  DE  TRAVAIL ***
 01 LIG1.
     02 NOM-ETUD1 PIC X(20).
     02 PRENOM-ETUD1 PIC X(15).
 01 LIGNE1.
     02 FILLER PIC X(400).
     02 ENONCE-ITEM PIC X(140).
 01 LIGNE2.
     02 FILLER PIC XX.
     02 REA-E2 COMP PIC 99.
     02 NOM2       PIC X(20).
     02 PRENOM2    PIC X(15).
     02 ADRESSE2.
        03 NUM2    PIC X(4).
        03 VOI2    PIC X(30).
        03 COM2    PIC X(20).
        03 COD2    PIC X(5).
        03 BUR2    PIC X(20).
     02 CONTEXTE2 PIC X(20).
 01 ADRESSE-ETUD.
     02 NUM-ETUD PIC X(4).
     02 VOIE-ETUD PIC X(30).
     02 COM-ETUD PIC X(20).
     02 COD-ETUD PIC X(5).
     02 BUR-ETUD PIC X(20).
 01 LIGNE3 .
     02 FILLER PIC XX.
     02 REA-E3 COMP PIC 99.
     02 FILLER PIC X(396).
     02 NOM3  PIC X(20).
     02 PRENOM3 PIC X(15).
     02 ADRESSE3.
        03 NUM3 PIC X(4).
        03 VOI3 PIC X(30).
        03 COM3 PIC X(20).
        03 COD3 PIC X(5).
        03 BUR3 PIC X(20).
     02 CONTEXTE3 PIC X(20).
```



```
 01 LIGNE7.
     02 FILLER PIC XX.
     02 REA-E7 COMP PIC 99.
 01 TAB-CONC VALUE "C1 C2 C3 C4 C5 C6 C7 C8 C9 C10".
     02 NOM-CONC OCCURS 10 TIMES PIC X(3).
 01 LIGNE4.
     02 FILLER PIC XX.
     02 CONTENU COMP PIC 99.
     02 FILLER PIC XX.
     02 I4 COMP PIC 99.
 01 LIGNE5.
     02 FILLER PIC XX.
     02 EXISTE1 COMP PIC 99.
     02 NOM-CONT5 PIC X(20).
 01 LIGNE6.
     02 FILLER PIC X(8).
     02 REA-E6 COMP PIC 99.
     02 NOM-VERS6 PIC X.
     02 NOM-CONT6 PIC X(20).
 01 LIGNE8.
     02 NOM-VERS8  PIC X.
     02 NOM-CONT8  PIC X(20).
     02 FILLER PIC X(389).
     02 ENONCE8 PIC X(140).
 01 LIGNE9.
     02 FILLER PIC XX.
     02 DEBUT COMP PIC 99.
     02 FILLER PIC X(4).
     02 NOM-LEC9 PIC X(20).
 01 LIGNE10.
     02 NOM-CONT10 PIC X(20).
 01 TAB-CONT1.
     02 TAB-CONT OCCURS 3 TIMES  PIC X(20).
 PROCEDURE DIVISION.
 DECLARATIVES.
 SEC1 SECTION.
      USE AFTER STANDARD ERROR PROCEDURE ON F-LECT TOTO F-IMPR.
 ATU.    GO TO FERMETURE.
 END DECLARATIVES.
*                    ****  PROLOGUE  ****
 SEC2 SECTION.
 OUVERTURES.
     OPEN OUTPUT F-IMPR.
     OPEN INPUT F-LECT.
*    OPEN I-O TOTO
     MOVE "U" TO MDR.
     MOVE "O" TO SCR.
     CALL SINIT USING NCD LCD NSI NBR LCP NCP MDR SCR COR.
     MOVE "U" TO MDS.
     MOVE "U" TO TYP.
     CALL SOPEN USING TABLE-INTERFACE MDS TYP.
 DEBUT.
*    MOVE UN TO CLET.
*    WRITE ARTICLE1 INVALID KEY PERFORM CPAUSE.
*    GO TO NORM.
* CPAUSE. MOVE DEUX TO CLET.
*    READ TOTO RECORD INVALID KEY
*                        GO TO FERMETURE.
*    IF MODE-EN-COURS = "T" MOVE "O" TO PAUSE
*                                GO TO TUTORIEL.
*    IF MODE-EN-COURS = "A" MOVE "O" TO PAUSE
*                              GO TO ADAPTATION.
*    GO TO FERMETURE.
```

```
*                 ****  PROLOGUE NORMAL  ****
 NORM. DISPLAY "Bonjour! Le logiciel de dialogue SATIRE est à votr
-     "e service. Est-ce la première fois que vous l'utilisez?".
      READ F-LECT.
      IF REP1 = "OUI" PERFORM CREATION-ETUDIANT THRU F-CREATION
                      ELSE PERFORM VERIFICATION-IDENTITE THRU F-VERIF
-     .
      MOVE NOM-ETUD TO NOM-T.
      MOVE PRENOM-ETUD TO PRENOM-T.
      MOVE NUM-REA-ETUD TO REA-E7.
      CALL SGBD USING TABLE-INTERFACE LIGNE7 LONG7
              UN INI-ETUD
              UN VAR-X1.
      DISPLAY "Voulez-vous qu'on vous rappelle les possibilités de
-     "SATIRE ?".
      READ F-LECT.
      IF REP1 = "OUI" PERFORM ITEM-POSSI.
      MOVE "P" TO MOD-PREC.
      PERFORM AIGUILLAGE.
*               ***  FIN DU PROLOGUE   ***
 CREATION-ETUDIANT.
      DISPLAY "Nous allons vous demander quelques renseignements su
-     "r votre identité.Votre nom:".
      READ F-LECT INTO NOM-ETUD.
      DISPLAY "Votre prénom:".
      READ F-LECT INTO PRENOM-ETUD.
      PERFORM RECH-ETUDC THRU F-RECH-ETUD.
      IF EXISTE2 = "N" PERFORM RENS
                       PERFORM CREAT-ETUDC THRU F-CRET
                               GO TO F-CREATION
              ELSE     PERFORM SCREAT GO TO F-CREATION.
 SCREAT.
      DISPLAY "Il existe déjà dans la base un étudiant de nom:"
-     NOM-ETUD "et de prénom :" PRENOM-ETUD.
      DISPLAY "Son adresse est :" ADRESSE-ETUD .
      DISPLAY "Est-ce-vous?".
      READ F-LECT.
      IF REP1 = "NON"
                 PERFORM RECH-ETUDCS THRU F-RECH-ETUDS
                 IF EXISTE2 NOT = "N" GO TO SCREAT
                            ELSE PERFORM RENS
                                 PERFORM CREAT-ETUDC THRU F-CRET
                ELSE MOVE REA-E2 TO NUM-REA-ETUD.
 F-CREATION.
      EXIT.
 VERIFICATION-IDENTITE.
      MOVE 1 TO I.
      DISPLAY "Nous allons vous identifier.".
 DE1. IF I > 3 GO TO ABANDON.
      DISPLAY "Votre nom: ".
      READ F-LECT INTO NOM-ETUD.
      DISPLAY "Votre prénom: ".
      READ F-LECT INTO PRENOM-ETUD.
      PERFORM RECH-ETUDC THRU F-RECH-ETUD.
 PH1. IF EXISTE2 = "N" PERFORM VI1
                       ELSE PERFORM VI2.
 F-VERIF.
      EXIT.
 VI2.
      DISPLAY "Etes-vous bien " NOM-ETUD PRENOM-ETUD.
      DISPLAY ADRESSE-ETUD.
      READ F-LECT.
      IF REP1 = "NON" PERFORM RECH-ETUDCS THRU F-RECH-ETUDS
                      GO TO PH1
                ELSE MOVE REA-E2 TO NUM-REA-ETUD.
```



```
 VI1.
      DISPLAY "Vous etes-vous trompé dans votre nom?".
      READ F-LECT.
      IF REP1 = "OUI" ADD UN I GO TO DE1
                   ELSE PERFORM RENS
                        PERFORM CREAT-ETUDC THRU F-CRET.
*            ***      AIGUILLAGES          ***
 AIGUILLAGE.
      DISPLAY "          ****************************************".
      DISPLAY "Choisissez votre mode en tapant une des commandes :
-     "A,D,T,R,FIN".
      READ F-LECT INTO COM-SEC.
      IF COM-SEC = "A" GO TO ADAPTATION.
      IF COM-SEC = "D" GO TO DOCUMENTAIRE.
      IF COM-SEC = "T" GO TO TUTORIEL.
      IF COM-SEC = "R" GO TO RECAPITULATIF.
      IF COM-SEC = "FIN" GO TO FERMETURE.
      IF COM-SEC = "RET" GO TO NORM
                         ELSE GO TO AIGUILLAGE.
 ADAPTATION.
      GO TO FERMETURE.
 TUTORIEL.
      GO TO FERMETURE.
 RECAPITULATIF.
      GO TO FERMETURE.
 ABANDON.
      GO TO FERMETURE.
 ERREUR.
      GO TO FERMETURE.
 FERMETURE.
      CALL SCLOSE USING TABLE-INTERFACE.
      CALL SFINISH.
      CLOSE F-IMPR F-LECT .
*     CLOSE TGTO .
      GO TO FINPROG.
 ITEM-POSSI.
      CALL SGBD USING TABLE-INTERFACE LIGNE1 LONG1
         UN ITEM-POSS
         UN VAR-X1.
      IF CODE-ANOMALIE NOT = ZERO PERFORM ERREUR.
      WRITE LIGNE FROM ENONCE-ITEM.
*                ***  SS-PGME  RECHERCHE L"EXISTENCE D"1 ETUDIANT +
*     ADRESSE
 RECH-ETUDC.
      MOVE NOM-ETUD TO NOM2.
      MOVE PRENOM-ETUD TO PRENOM2.
      CALL SGBD USING TABLE-INTERFACE LIGNE2 LONG2
         UN RECH-ETUD
         UN VAR-X1.
      IF CODE-ANOMALIE NOT = ZERO PERFORM ERREUR.
      IF REA-E2 = 0 MOVE "N" TO EXISTE2
              GO TO F-RECH-ETUD
          ELSE MOVE "O" TO EXISTE2
      MOVE ADRESSE2 TO ADRESSE-ETUD.
      MOVE CONTEXTE2 TO CONTEXTE-ETUD.
 F-RECH-ETUD.
      EXIT.
```

```
*                *** RECUPERE L"ADRESSE DE  L"ETUDIANT ***
 RENS.
     DISPLAY "Votre adresse(numéro,voie,commune,code-postal,bureau
-     " distributeur).".
     DISPLAY "Numéro:".
     READ F-LECT INTO NUM-ETUD.
     DISPLAY "Voie: ".
     READ F-LECT INTO VOIE-ETUD.
     DISPLAY "Commune: ".
     READ F-LECT INTO COM-ETUD.
     DISPLAY "Code Postal: ".
     READ F-LECT INTO COD-ETUD.
     DISPLAY "Bureau Distributeur: ".
     READ F-LECT INTO BUR-ETUD.
     DISPLAY "Les informations disponibles dans la banque de donné
-    "es de SATIRE s'adressent à certains publics. Nous souhaitons
-    " savoir si    vous appartenez à une de ces catégories.".
     DISPLAY "Les catégories prévues actuellement,que nous appelon
-    "s des contextes,sont écrites dans la notice. Veuillez les co
-    "nsulter et     tapez ensuite le nom du contexte que vous choi
-    "isissez".
     READ F-LECT INTO CONTEXTE-ETUD.
*
 RECH-ETUDCS.
     MOVE NOM-ETUD TO NOM2.
     MOVE PRENOM-ETUD TO PRENOM2.
     CALL SGBD USING TABLE-INTERFACE LIGNE2 LONG2
        UN RECH-ETUDS
        DEUX VAR-X1 VAR-X2.
     IF CODE-ANOMALIE NOT = ZERO PERFORM ERREUR.
     IF REA-E2 = ZERO MOVE "N" TO EXISTE2
              ELSE MOVE "O" TO EXISTE2
                   MOVE ADRESSE2 TO ADRESSE-ETUD.
             MOVE CONTEXTE2 TO CONTEXTE-ETUD.
 F-RECH-ETUDS.
     EXIT.
*                 ***  CREATION D"UN ETUDIANT DANS LA BASE  ***
 CREAT-ETUDC.
     MOVE NOM-ETUD TO NOM3.
     MOVE PRENOM-ETUD TO PRENOM3.
     MOVE ADRESSE-ETUD TO ADRESSE3.
     MOVE CONTEXTE-ETUD TO CONTEXTE3.
     CALL SGBD USING TABLE-INTERFACE LIGNE3 LONG3
        UN CREAT-ETUD
        UN VAR-X1.
     IF CODE-ANOMALIE NOT = ZERO PERFORM ERREUR.
     IF REA-E3 = ZERO DISPLAY "Il n'y a plus de place pour un nou
-    "vel étudiant. Veuillez avertir l'administrateur de la base
-    "et mettez-vous d'accord avec lui pour un nouveau rendez-vou
-    "s."
                      GO TO ABANDON
          ELSE MOVE REA-E3 TO NUM-REA-ETUD.
 F-CRET.  EXIT.
*                        *** MODE DOCUMENTAIRE  ***
 DOCUMENTAIRE.
     IF MOD-PREC = "A" DISPLAY "Vous vous etes sans doute trompé
-    " en demandant ce mode. Voici les diverses possibilités de S
-    "ATIRE :"
                       PERFORM ITEM-POSSI
                       MOVE "D" TO MOD-PREC
                       PERFORM AIGUILLAGE.
```



```
 DO1.DISPLAY "Tapez le numéro de l'information à laquelle vous vou
-    "lez accéder(liste des informations dans la notice):".
 DO4.   READ F-LECT INTO ICONC.
     MOVE ICONC TO I4.
     IF I4 < 1 OR I4 > 10 DISPLAY "Il n'y a pas d'information repé
-    "rée par ce numéro. Recommencez:" GO TO DO4.
     CALL SGBD USING TABLE-INTERFACE LIGNE4 LONG4
          UN RECH-CONC
          DEUX VAR-X1 VAR-X2.
     IF CONTENU NOT = ZERO PERFORM SUIT-DOC THRU F-S-DOC
            ELSE PERFORM PREREQUIS THRU F-PRER.
     DISPLAY "Si vous désirez sortir du mode documentaire, tapez B
-    "YE sinon appuyez sur la touche RETURN".
     READ F-LECT INTO COM-SEC.
     IF COM-SEC = "BYE" PERFORM AIGUILLAGE
                        ELSE GO TO DO1.
 PREREQUIS.
     CALL SGBD USING TABLE-INTERFACE LIGNE7 LONG7
          UN MAJ-PRER
          DEUX VAR-X1 VAR-X2.
     DISPLAY "La base ne contient pas d'information sur cette noti
-    "on, qui est supposée connue.Nous avons cependant pris note d
-    "e votre         demande. Excusez-nous.".
 F-PRER. EXIT.
 SUIT-DOC.
     MOVE CONTEXTE-ETUD TO NOM-CONT5.
     CALL SGBD USING TABLE-INTERFACE LIGNE5 LONG5
            UN RECH-CONT
            DEUX VAR-X1 VAR-X2.
     IF EXISTE1 = ZERO PERFORM CHOIX-CONT THRU F-CONT
               ELSE MOVE CONTEXTE-ETUD TO NOM-CONT.
     MOVE NUM-REA-ETUD TO REA-E6.
     MOVE NOM-CONT TO NOM-CONT6.
     CALL SGBD USING TABLE-INTERFACE LIGNE6 LONG6
            UN RECH-VERS
            QUAT VAR-X1 VAR-X2 VAR-X3 VAR-X4.
     DISPLAY NOM-VERS6.
*      ***  PRESENTATION DE  L ITEM   ***
*     MOVE NOM-VERS6 TO NOM-VERS8.
*    MOVE NOM-CONT TO NOM-CONT8.
*    CALL SGBD USING TABLE-INTERFACE LIGNE8 LONG8
*            UN PRES-ITEM
*            TROIS VAR-X1 VAR-X2 VAR-X3.
*    WRITE LIGNE FROM ENONCE8.
*      ***  PRESENTATION DES LECONS  ***
     DISPLAY "Voulez-vous connaitre les cours  dont ce concept fai
-    "t partie?".
     READ F-LECT.
     IF REP1 = "OUI" PERFORM LECONS THRU F-LEC.
 F-S-DOC. EXIT.
 LECONS.
     DISPLAY "Ce concept fait partie des cours suivants:".
     MOVE "   " TO NOM-LEC9.
     MOVE ZERO TO DEBUT.
 DO2.
     CALL SGBD USING TABLE-INTERFACE LIGNE9 LONG9
              UN LEC-CONC
              DEUX VAR-X1 VAR-X2.
     IF NOM-LEC9 NOT = "FIN" DISPLAY NOM-LEC9
                             GO TO DO2.
 F-LEC. EXIT.
```

```
*              ***  CHOIX DU  CONTEXTE   ***
 CHOIX-CONT.
     CALL SGBD USING TABLE-INTERFACE LIGNE4 LONG4
             UN NBRE-CONT
             TROIS VAR-X1 VAR-X2 VAR-X3.
     DISPLAY "L'information ne figure pas dans le contexte que vou
-    "s avez choisi. Voici les contextes disponibles:".
     MOVE UN TO I.
 DO3.
      CALL SGBD USING TABLE-INTERFACE LIGNE10 LONG10
              UN PRES-CONT
              TROIS VAR-X1 VAR-X2 VAR-X3.
     MOVE NOM-CONT10 TO TAB-CONT (I).
     MOVE I TO ICONT.
     DISPLAY ICONT NOM-CONT10.
     ADD UN TO I.
     IF I NOT > CONTENU GO TO DO3.
     DISPLAY "Tapez le numéro correspondant au contexte choisi.".
     READ F-LECT INTO ICONT.
     MOVE ICONT TO I.
     MOVE TAB-CONT (I) TO NOM-CONT.
 F-CONT. EXIT.
*                                          ***
 FINPROG.  STOP RUN.
!ASSIGN GO,FIL,(STS,OLD),(NAM,SOCBO2),(UNT,AC,INTG)
!ASSIGN TREE,FIL,(STS,OLD),(NAM,SOCV2-SOCTREA),(UNT,AC,:SYS)
!ASSIGN LM,FIL,(SIZ,(RST),2),(NAM,LM-SATIRE)
!ASSIGN BIB,FIL,(STS,OLD),(NAM,COBLIB),(UNT,AC,:SYS)
!LINK
:OPTION (UNSAT,BIB)
:TREE,FIL
```

8.5.2. *Extrait des sous-programmes SOCRATE utilisés.*

```
 F/CREAT-ETUD/
00001*1,$L
00001  C(0) UN ETUDIANT X1
00002 SI SYS-ANOM = X'06D7' ALORS M Y1 = 0 SINON
00003  M Y1 = NUMDE X1
00004  FIN
FIN DE FICHIER
*
```

```
 F/INI-ETUD/
00001*1,$L
00001 M X1 = UN ETUDIANT Y1
FIN DE FICHIER
*
```



```
 PRECOMP
*F/RECH-ETUD/
00001*1,$L
00001 D YZO = DEBUT
00002          ESPACE 4
00003          NOM MOT 20
00004          PREN MOT 15
00005          ADRESSE DEBUT
00006             NUM MOT 4
00007             VOIE MOT 30
00008             COMMUNE MOT 20
00009             CODEPOS MOT 5
00010             BUREAU MOT 20
00011           FIN
00012          CON MOT 20
00013 FIN
00014 SI EXISTE UN ETUDIANT AYANT NOM = NOM DE YZO
00015     ET PRENOM = PREN DE YZO ALORS
00016  M X1 = UN ETUDIANT AYANT NOM = NOM DE YZO ET
00017     PRENOM = PREN DE YZO
00018 POUR YZO
00019 M NUM DE ADRESSE = NUMERO DE ADRESSE DE X1
00020 M VOIE DE ADRESSE = VOIE DE ADRESSE DE X1
00021 M COMMUNE DE ADRESSE = COMMUNE DE ADRESSE DE X1
00022 M CODEPOS DE ADRESSE = CODE-POSTAL DE ADRESSE DE X1
00023 M BUREAU DE ADRESSE = BUREAU-DISTRIBUTEUR DE ADRESSE DE X1
00024 M CON = CONTEXTE DE X1
00025 FIN
00026 M Y1 = NUMDE X1
00027     SINON
00028 M SYS-ANOM = U
00029 M Y1 = 0
00030   FIN
FIN DE FICHIER
*
```

```
 F/LEC-CONC/
00001*1,$L
00001 D YZO = DEBUT
00002       ESPACE 8
00003       Z-LEC MOT 20
00004 FIN
00005 D X2 = UN CONCEPT
00006 M Y1 = Y1 + 1
00007 SI Y1 > NBDE TOUTE LECON ALORS M Z-LEC = 'FIN'
00008                              SINON
00009      SI NUM-LECON DE ETUDIE-DANS (Y1) DE X2 = U
00010       ALORS M Z-LEC = 'FIN'
00011 SINON
00012      M Y2 = NUM-LECON DE ETUDIE-DANS (Y1) DE X2
00013                 M Z-LEC = NOM DE UNE LECON Y2
00014      FIN
00015 FIN
FIN DE FICHIER
*
```

```
 F/RECH-ETUDS/
00001*1,$L
00001 D YZ0 = DEBUT
00002          ESPACE 4
00003          Z-NOM MOT 20
00004          Z-PREN MOT 15
00005          Z-ADR 79
00006          Z-CON MOT 20
00007          FIN
00008  D Z-ADR = DEBUT
00009           Z-NUM MOT 4
00010           Z-VOI MOT 30
00011           Z-COM MOT 20
00012           Z-COD MOT 5
00013           Z-BUR MOT 20
00014           FIN
00015 M Y1 = 1
00016 D X1 = UN ETUDIANT
00017 FAIRE
00018      M X1 = SUIVANT DE X1
00019     SI SYS-ANOM = X'01D7' ALORS M Y1 = 0 SORTIE SINON
00020           SI NOM DE X1 = Z-NOM ET PRENOM DE X1 = Z-PREN
00021                           ALORS SORTIE
00022                           SINON REFAIRE
00023           FIN
00024   FIN
00025 FIN
00026  SI Y1 = 1 ALORS
00027 M Y1 = NUMDE X1
00028 M Z-ADR = U
00029 M Z-NUM = NUMERO DE ADRESSE DE X1
00030 M Z-VOI = VOIE DE ADRESSE DE X1
00031 M Z-COM = COMMUNE DE ADRESSE DE X1
00032 M Z-COD = CODE-POSTAL DE ADRESSE DE X1
00033 M Z-BUR = BUREAU-DISTRIBUTEUR DE ADRESSE DE X1
00034 M Z-CON = CONTEXTE DE X1
00035  FIN
FIN DE FICHIER
*
```

```
 F/PRES-CONT/
00001*1,$L
00001 D YZ0 = DEBUT
00002     Z-CONT MOT 20
00003 FIN
00004 D X2 = UN CONCEPT
00005 M X3 = SUIVANT DE UN CONTEXTE DE X2
00006 M Z-CONT = NOM DE X3
FIN DE FICHIER
*
```

```
 F/NBRE-CONT/
00001*1,$L
00001 D X2 = UN CONCEPT
00002 M Y1 = NBDE TOUT CONTEXTE DE X2
00003 M X3 = U
FIN DE FICHIER
*
```



```
 F/RECH-CONT/
00001*1,$L
00001 D YZ0 = DEBUT
00002     ESPACE 4
00003     Z-CONT MOT 20
00004     Z-CONC MOT 40
00005 FIN
00006 D X2 = UN CONCEPT
00007 M Z-CONC = NOM DE X2
00008 ECRIRE Z-CONC
00009 SI EXISTE UN CONTEXTE AYANT NOM = Z-CONT  DE X2
00010                     ALORS M Y1 = 1 ECRIRE Z-CONT
00011                     SINON M Y1 = 0
00012 FIN
FIN DE FICHIER
*
```

```
 F/RECH-VERS/
00001*1,$L
00001 D YZ0 = DEBUT
00002     ESPACE 8
00003     Z-ETUD BINAIRE 2
00004     Z-NOM MOT 1
00005     Z-CONT MOT 20
00006 FIN
00007 D X2 = UN CONCEPT
00008 M Y2 = 0
00009 FAIRE
00010 M Y2 = Y2 + 1
00011 SI NUM-ETUDIANT DE ETUDIE-PAR (Y2) DE X2 = U
00012  ALORS SORTIE
00013 SINON
00014 M Y1 = NUM-ETUDIANT DE ETUDIE-PAR (Y2) DE X2
00015            SI Y1 = Z-ETUD ALORS SORTIE
00016                                SINON REFAIRE
00017                  FIN
00018            FIN
00019 FIN
00020 M X3 = UN CONTEXTE AYANT NOM = Z-CONT DE X2
00021 SI NUM-ETUDIANT DE ETUDIE-PAR (Y2) DE X2 = U
00022    ALORS
00023    M X4 = UNE VERSION 2 DE X3
00024    M Z-NOM = NOM DE X4
00025            SINON
00026    M Z-NOM = VERSION-A-PRESENTER DE ETUDIE-PAR (Y2) DE X2
00027    M X4 = UNE VERSION AYANT NOM = Z-NOM DE X3
00028 FIN
00029 M X4 = SUIVANT DE X4
00030 SI SYS-ANOM = X'01D7'
00031                ALORS M X4 = UNE VERSION 2 DE X3
00032                SINON M Y25 = U
00033 FIN
00034 POUR X2
00035    M VERSION-A-PRESENTER DE ETUDIE-PAR (Y2) = NOM DE X4
00036    M NUM-ETUDIANT DE ETUDIE-PAR (Y2) = Z-ETUD
00037 FIN
FIN DE FICHIER
*
```

```
 F/MAJ-PRER/
00001*1,$L
00001 D X2 = UN CONCEPT
00002 G UN PREREQUIS-DEMANDE = X2
FIN DE FICHIER
*
```

```
 F/RECH-CONC/
00001*1,$L
00001    M X2 = UN CONCEPT Y2
00002  SI ANTECEDENTS (1) DE X2 = U
00003                    ALORS M Y1 = 0
00004                    SINON M Y1 = 1
00005 FIN
FIN DE FICHIER
*
```

8.5.3. *Extrait d'un dialogue.*

```
satire
jour
1505 060P00 00/000/102
Bonjour! Le logiciel de dialogue SATIRE est à votre service. Est-ce la première fois que vous l'utilisez?
?OUI
Nous allons vous demander quelques renseignements sur votre identité.Votre nom:
?SCHUSTER
Votre prénom:
?ELISABETH
Votre adresse(numéro,voie,commune,code-postal,bureau distributeur).
Numéro:
?6
Voie:
?RUE DE L HERMINE
Commune:
?RENNES
Code Postal:
?35000
Bureau Distributeur:
?RENNES
Les informations disponibles dans la banque de données de SATIRE s'adressent à certains publics. Nous souhaitons savoir si
vous appartenez à une de ces catégories.
Les catégories prévues actuellement,que nous appelons des contextes,sont écrites dans la notice. Veuillez les consulter et
tapez ensuite le nom du contexte que vous choisissez
?SPECIALISTE
Voulez-vous qu'on vous rappelle les possibilités de SATIRE ?
?NON
                    ************************************
Choisissez votre mode en tapant une des commandes : A,D,I,R,FIN
?D
Tapez le numéro de l'information à laquelle vous voulez accéder(liste des informations dans la notice):
?01
1
Voulez-vous connaitre les cours  dont ce concept fait partie?
?OUI
Ce concept fait partie des cours suivants:
LA PROGRAMMATION
L ANALYSE
Si vous désirez sortir du mode documentaire, tapez BYE sinon appuyez sur la touche RETURN
?
Tapez le numéro de l'information à laquelle vous voulez accéder(liste des informations dans la notice):
?02
La base ne contient pas d'information sur cette notion, qui est supposée connue.Nous avons cependant pris note de votre
demande. Excusez-nous.
Si vous désirez sortir du mode documentaire, tapez BYE sinon appuyez sur la touche RETURN
?
Tapez le numéro de l'information à laquelle vous voulez accéder(liste des informations dans la notice):
?01
1
Voulez-vous connaitre les cours  dont ce concept fait partie?
?NON
Si vous désirez sortir du mode documentaire, tapez BYE sinon appuyez sur la touche RETURN
?BYE
                    ************************************
Choisissez votre mode en tapant une des commandes : A,D,I,R,FIN
?RET
```



```
Bonjour! Le logiciel de dialogue SATIRE est à votre service. Est-ce la première fois que vous l'utilisez?
?OUI
Nous allons vous demander quelques renseignements sur votre identité.Votre nom:
?SCHUSTER
Votre prénom:
?ELISABETH
Il existe déjà dans la base un étudiant de nom: SCHUSTER              et de prénom : ELISABETH
Son adresse est : 6   RUE DE L HERMINE              RENNES              35000RENNES
Est-ce-vous?
?NON
Votre adresse(numéro,voie,commune,code-postal,bureau distributeur).
Numéro:
?15
Voie:
?RUE MAILLANE
Commune:
?ST-AVOLD
Code Postal:
?57500
Bureau Distributeur:
?ST-AVOLD
Les informations disponibles dans la banque de données de SATIRE s'adressent à certains publics. Nous souhaitons savoir si
vous appartenez à une de ces catégories.
Les catégories prévues actuellement,que nous appelons des contextes,sont écrites dans la notice. Veuillez les consulter et
tapez ensuite le nom du contexte que vous choisissez
?LITTERAIRE
Voulez-vous qu'on vous rappelle les possibilités de SATIRE ?
?NON
                    ************************************
Choisissez votre mode en tapant une des commandes : A,D,I,R,FIN
?D
Tapez le numéro de l'information à laquelle vous voulez accéder(liste des informations dans la notice):
?03
L'information ne figure pas dans le contexte que vous avez choisi. Voici les contextes disponibles:
1 SPECIALISTE
Tapez le numéro correspondant au contexte choisi.
?1
1
Voulez-vous connaitre les cours  dont ce concept fait partie?
?OUI
Ce concept fait partie des cours suivants:
LA PROGRAMMATION
L ANALYSE
Si vous désirez sortir du mode documentaire, tapez BYE sinon appuyez sur la touche RETURN
?BYE
                    ************************************
Choisissez votre mode en tapant une des commandes : A,D,I,R,FIN
?RET
```

```
Bonjour! Le logiciel de dialogue SATIRE est à votre service. Est-ce la première fois que vous l'utilisez?
?NON
Nous allons vous identifier.
Votre nom:
?BERTAUD
Votre prénom:
?ANTOINE
Vous etes-vous trompé dans votre nom?
?OUI
Votre nom:

Votre prénom:
?FRANCOIS
Vous etes-vous trompé dans votre nom?
?NON
Votre adresse(numéro,voie,commune,code-postal,bureau distributeur).
Numéro:
?65
Voie:
?RUE ANATOLE FRANCE
Commune:
?NANCY
Code Postal:
?54000
Bureau Distributeur:
?NANCY
Les informations disponibles dans la banque de données de SATIRE s'adressent à certains publics. Nous souhaitons savoir si
vous appartenez à une de ces catégories.
Les catégories prévues actuellement,que nous appelons des contextes,sont écrites dans la notice. Veuillez les consulter et
tapez ensuite le nom du contexte que vous choisissez
?SCIENTIFIQUE
Voulez-vous qu'on vous rappelle les possibilités de SATIRE ?
?NON
                    ************************************
Choisissez votre mode en tapant une des commandes : A,D,I,R,FIN
?D
Tapez le numéro de l'information à laquelle vous voulez accéder(liste des informations dans la notice):
?04
```

6.6. TRAITEMENT DES ALGORITHMES ECRITS SELON LA METHODE DE PROGRAMMATION DEDUCTIVE.

On a vu au paragraphe 6.4.1. que l'utilisation d'un terminal "intelligent" permettait d'imprimer les algorithmes construits grâce à la méthode de programmation déductive dans l'ordre de leur conception.

Ceci est réalisé de la façon suivante : quand le programme COBOL imprime le contenu d'une variable de type texte (buffer) et que cette variable contient le code hexadécimal d'une fonction du terminal, la fonction est exécutée, le reste du texte étant imprimé normalement.

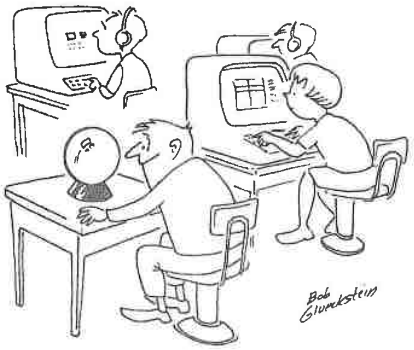
Il faut donc que les réalisations de l'entité ligne correspondant à l'énoncé d'un item contiennent de tels codes hexadécimaux. Se pose alors le problème de leur saisie. Il est bien sûr impensable que les enseignants introduisant les items connaissent de tels codes !

Nous avons donc décidé de définir un jeu de commandes mnémoniques, par exemple :

+ M4 pour "faire monter le papier de 4 interlignes".

Lors de l'introduction des items, ces commandes sont insérées dans le texte, un programme SOCRATE se chargeant alors de leur décodage et de leur traduction dans le ou les codes hexadécimaux des fonctions nécessaires pour leur exécution.

Une commande particulière correspond à l'interruption de l'impression. En effet, après impression d'une définition, il est souhaitable de laisser à l'étudiant le temps qu'il faut pour la comprendre. L'étudiant doit ensuite, quand il le désire, relancer l'impression. Les fonctions requises sont annulation de l'impression (pour éviter le ?), descente du papier de deux interlignes, rétablissement de l'impression.



CONCLUSION et PERSPECTIVES

Ce travail se situe à la charnière entre deux époques : l'époque qu'on pourrait appeler l'ère de l'enseignement assisté par ordinateur, et l'époque que d'autres que nous ont appelé celle de la télématique. Son intérêt est donc à prendre en considération de ces deux points de vue.

En ce qui concerne l'enseignement assisté par ordinateur, il apporte un certain recul sur la plupart des applications, réalise une synthèse et perfectionne certains outils. Le projet SATIRE, à la fois opérationnel et inopérant, a permis de tester la faisabilité de ces outils, dans un sens positif ou négatif, ce qui est tout aussi utile. On n'a pas recherché l'universalité, laissant de côté des domaines importants tels que la communication en langue naturelle, déjà explorés dans d'autres centres : on s'est au contraire attaché à aborder de façon méthodologique ce qui, sans être négligé, avait été vu de façon trop pragmatique parce que semblant aller de soi.

Il reste cependant deux directions de prolongements possibles, que nous avons signalées au cours de l'ouvrage.

La première concerne la prise en compte des erreurs commises par l'étudiant dans la fabrication d'algorithmes déductifs (chapitre 6). Nous avons dit que l'outil d'analyse reste à écrire. Ce travail sera peut-être fait dans le cadre de l'équipe CASTOR du CRIN. En liaison avec le système SATIRE, nous pourrions alors disposer d'un bon outil d'observation des conduites des étudiants lors de leur apprentissage de la programmation.

La seconde consiste en la définition, voire la création, de structures d'informations adéquates pour la mémorisation des données relatives à l'informatique éducationnelle (chapitre 7). Sans doute faudra-t-il le faire dans une optique de partage de tâches entre gros systèmes et micro-ordinateurs [190]. Un projet de réseau actuellement à l'étude à Nancy sera peut-être l'occasion de poursuivre ce genre de recherches, et me sert de transition pour la seconde partie de cette conclusion.

Les idées développées dans cette thèse auront-elles encore une quelconque utilité dans l'ère de la télématique ? Je le crois. Non seulement l'individualisation de l'enseignement est un phénomène qui ne peut que s'accentuer avec la prolifération des micro-ordinateurs et des réseaux de télécommunications, mais le droit de chacun à la formation va pousser l'enseignement hors de l'institution éducative : il est alors important que les concepteurs de didacticiels, quelle que soit leur origine, puissent bénéficier des résultats de recherches spécialisées dans le domaine.

Pour ma part, si je n'envisage pas de continuer mes recherches dans le prolongement direct de ce travail[1], je n'abandonnerai pas ce terrain passionnant de l'aide de l'informatique à l'enseignement, en profitant de l'expérience accumulée dans l'époque passée, et en intégrant les technologies de l'époque à venir.

(1) d'une part parce que je n'en ai pas les moyens matériels, d'autre part parce que, ayant souffert d'un certain isolement sur le plan nancéen, je souhaite me rapprocher de collègues d'autres équipes.

"Can I help it if it's programmed to teach, and I'm programmed to forget?"

BIBLIOGRAPHIE

Cette bibliographie contient :

- des références bibliographiques classiques (livres, articles, rapports),
- des ouvrages dont la lecture a éclairé l'environnement pédagogique et didactique du travail, sans qu'il y soit nécessairement fait référence dans le texte,
- des applications dont nous avons eu connaissance par des visites, des documents écrits, des groupes de travail ou des colloques : elles sont en général référencées par leur nom ou le nom de leur responsable principal, la citation de toutes les publications qui les concernent pouvant s'avérer fastidieuse.

[1] ADDA, J. Initiation au langage mathématique - Analyse d'une expérience d'enseignement. Bulletin "Didactique des Disciplines", Université de Paris VII, n° 4, juin 1975.

[2] ALLEN, M.W. Computer managed instruction : a definitive design. In [115], Part 1, pp 115-122.

[3] ANNETT, J. Psychological bases of educational technology. Paper to APLET, Brighton, 1973.

[4] ARBIB, M.A. Theories of abstract automata. Prentice Hall, 1969.

[5] ASET. UNIVAC.

[6] BACHELARD, G. La formation de l'esprit scientifique. Editions J. VRIN, Paris, 1938.

[7] BARRE, J ; COUVERT, A ; LE PALMEC, J et MONNIER, E. Manuel de référence du sous-ensemble PASCAL/IRIS 80. Publications de l'Université de Rennes 1, 1978.

[8] BARUK, S. Fabrice ou le procès mathématique. Le Seuil, Paris, 1977.

[9] BASIC sous SIRIS 7/SIRIS 8. Manuel d'utilisation et d'opérations. Documentation CII n° 4041 E1/FR, 1973.

[10] BELLEGARDE, F ; HUC, B ; PIERREL, J.M. et QUERE, A. Initiation à une construction méthodique des programmes. Publications du Centre de Recherche en Informatique de Nancy, n° 77-E-035, 1978.

[11] BELLISSANT, C. Teaching and learning languages. IFIP world conference on computer education, Amsterdam, 1970.

[12] BELLISSANT, C. Conception et compilation d'un langage pour l'écriture de cours. Thèse, Université de Grenoble, 1970.

[13] BENSMAINE, M. Contribution à l'étude du découpage et de la structuration d'une matière à enseigner ; expérimentation en E.A.O. Thèse, Université Paul Sabatier de Toulouse, 1974.

[14] BERGE, C. Graphes et hypergraphes. Dunod, Paris, 1970.

[15] BERGE, C. Optimisation d'une stratégie par la méthode des graphes. Les méthodes d'apprentissage. Working paper UCODI n° 1.

[16] BESTOUGEFF, H et al. Bilan Comparatif des expériences de rénovation pédagogique en enseignement assisté par ordinateur. Ministère des Universtés, 1979.

[17] BLOOM, B.S. et al. Taxonomie des objectifs pédagogiques. Education Nouvelle, Montréal, 1969.

[18] BONNET, A. Baobab, a parser for a rule-based system using a semantic grammar. Stanford Computer Science Dept, Report 668, 1978.

[19] BONSJAKOVIC, B. Fondements théoriques et empiriques de la longueur cohérente. Application à l'optimisation de l'enseignement de la physique et des mathématiques. Ecole d'été UCOOI, Genève, 1975.

[20] BOSSUET, G. Qu'est-ce que LOGO ? In [92].

[21] BOUSSARD, J.C. et LECARME, O. Didactique de la Programmation. Ecole d'été de l'AFCET, Montréal, 1977.

[22] BRIEN, R. Techniques de fabrication de cours. Publications du SIMEQ, Canada, 1971.

[23] British Airways Sales Training. Use of computer assisted teaching. Colloque européen sur l'enseignement assisté par ordinateur de l'Institut Européen pour la Formation Professionnelle, Monchy Saint Eloi, 1979.

[24] BROWN, J.S. ; BURTON, R.R. et ZOYBEL, F. A model-driven question-answering system for a CAI environment. Technical Report n° 13, Dept. of information and computer science, University of California, 1972.

[25] BUGGY. University of California, USA.

[26] BURKHARDT, D. A Multi-media approach for teaching computer science. Conseil de l'Europe, Conférence européenne sur le rôle et la valeur des nouvelles techniques de communication dans l'éducation post-secondaire, Strasbourg, 1979.

[27] BURTON, R.R. Semantic Grammar : an engineering technique for constructing natural language understanding systems. BBN Report n° 3453, 1976.

[28] CAMOL. London, U.K.

[29] CARBONELL, J.R. AI in CAI : an artificial-intelligence approach to computer-assisted instruction. IEEE Trans. Man-Mach. syst., vol MMS 11, 1970.

[30] CARDINET, J. Objectifs pédagogiques et fonction de l'évaluation. Publication de l'IREM d'Orléans, n° 7, 1977.

[31] CARMAN, R.A. Les vecteurs. Armand Colin, Paris, 1968.

[32] CARPEY, J.A.M. Activity algorithms and visual schemes in programmed instruction. Paderborner Arbeitspapier n° 27, 1976.

[33] CASCADE et ESPACE. Conception assistée par ordinateur, INPG, Grenoble, France.

[34] CASTELLANI, X. Organisation assistée d'un enseignement modulaire. Thèse, Université de Grenoble, 1975.

[35] CAUSSE, B. L'enseignement par calculateur d'un langage de programmation. In Enseignement Programmé, Dunod, Paris, n° 1, 1968.

[36] CHEIN, M. Aspects mathématiques des problèmes suivants : quel ordre choisir pour différentes parties d'un cours ou pour une batterie de tests. Ecole d'été UCOOI, Genève, 1975.

[37] CHERBONNEAU, B ; GALINIER, M ; LAGASSE, J.P. ; MASSIE, H ; MATHIS, B et PAUL, J.L. Programmation structurée. Ecole d'été de l'AFCET, Rabat, 1975.

[38] CIPOIRE, M ; CLOUZOT, O ; GOUIN, I ; QUERE, M et VIALLET, F. Un animateur, un groupe et un livre, ou les problèmes posés par l'insertion d'un document conçu pour les adultes dans une situation pédagogique : la P.S.T. Document interne CUCES, Nancy, 1969.

[39] CLOUZOT, O. La méthode d'analyse sémantique d'un contenu. In Enseignement programmé, n° 6, 1969.

[40] COLMERAUER, A. Un sous-ensemble intéressant du français. In [93].

[41] COLMEZ, F ; DELACOTE, G. et RICHARD, J.F. Statut de l'observation et de l'activité expérimentale chez l'élève. Table ronde "Didactique des sciences et psychologie", Paris, 1977.

[42] COULON, D. Conception et réalisation d'un programme d'enseignement assisté. Thèse, Faculté des sciences de Paris, 1970.

[43] COULON, D. Construction expérimentale d'un modèle informatique pour interprédes énoncés rédigés en langage naturel. Thèse, Université de Paris VI, 1975.

[44] COULON, D et KAYSER, D. Contribution de l'informatique à l'évolution des méthodes d'éducation. Publications de l'Institut de Programmation, Université de Paris VI, n° IP 73.8, 1973.

[45] COULON, D et KAYSER, D. Modalités du dialogue en enseignement assisté par ordinateur. In [92].

[46] COURSEWRITER et CMEAO. IBM.

[47] CREHANGE, M. Description formelle, représentation, interrogation des informations complexes. Système PIVOINES. Thèse, Université de Nancy 1, 1975.

[48] DARCHE, M et MONSELLIER, P. Pour une pédagogie par objectifs en mathématiques. Publications de l'IREM d'Orléans, n° 1, 1976.

[49] DEMARNE, P. Programme en "dents de scie" et spatialisation des concepts d'enseignement. In Enseignement Programmé, Dunod, Paris, n° 6, 1969.

[50] DENIS, J.P. et LOMBAERDE, J. Notes à propos des conclusions de l'Ecole d'été sur l'enseignement des sciences assisté par ordinateur, Louvain la Neuve, 1976.

[51] DERNIAME, J.C. et FINANCE, J.P. Types abstraits de données : spécification, utilisation et réalisation. Ecole d'été de l'AFCET, Monastir, 1979.

[52] DIGE. Progetto C.A.I., C.N.R., Genova, Italie.

[53] DONID, J. Contribution à l'analyse statistique et informatique des modèles stochastiques : application à l'apprentissage. Thèse, Université Paul Sabatier de Toulouse, 1971.

[54] EAO 5. Université de Paris V et CITI 2, France.

[55] Editeur de textes SIRIS 7/SIRIS 8 Temps Partagé. Versions B09/C09. Manuel d'utilisation. Documentation CII n° 4504E/FR, 1975.

[56] ELIZA. MIT, Cambridge, USA.

[57] ENSPI. ENS de Saint-Cloud, France.

[58] EPMINE et SOMINE. Ecole Nationale Supérieure des Mines de Saint-Etienne, France.

[59] FAFIOTTE, G. Influence de la pratique de l'E.A.O. sur l'évolution des logiciels d'aide à l'enseignement et sur la formation des enseignants. Colloque Franco-soviétique, 1975. In Bulletin de Psychologie, n° 330, 1976-77.

[60] FINANCE, J.P. ; HUC, B ; LESCANNE, P ; PAIR, C ; QUERE, M et REMY, J.L. Programmation déductive et structures de données. Publications du Centre de Recherche en Informatique de Nancy, n° 79-E-051, 1979.

[61] FISCHER, P et MERCIER, P. Langage LSE sous SIRIS7/SIRIS 8. Manuel d'utilisation. Notice de l'IUCAL n° NOTC 0031, 1976.

[62] FISZER, J. Un voyage d'étude aux Etats-Unis : l'enseignement assisté par ordinateur. In Enseignement Programmé, Dunod, Paris, n° 7, 1969.

[63] FRANK, H ; LANSKY, M et PROCHNOW, D. Lehrmaschinen in kybernetischer und pädagogischer Sicht. Klett-Oldenburg, 1964, 1965, 1966.

[64] GAGNE, R. The condition of learning. Rinehart et Winston, New York, 1965.

[65] GAVINI, G.P. Manuel de formation aux techniques de l'enseignement programmé. Paris, Hommes et Techniques, 1965.

[66] GERBIER, A. Mes premières constructions de programmes. Springer Verlag, 1977.

[67] GLAESER, G. La didactique de l'analyse. Non publié.

[68] GORALSKI, A. Heuristique. Colloque international "Langage et pensée mathématique", Luxembourg, 1976.

[69] GOUSTARD, M ; GRECO, M ; MATALON, B et PIAGET, J. La logique des apprentissages. PUF, Paris, 1959.

[70] GURBUTT, P. Quelques idées sur l'utilisation des graphes en physique. Working paper UCODI n° 14.

[71] HALBWACHS, F ; ROUCHIER, A et VERGNAUD, G. Structure de la matière enseignée, histoire des sciences et développement conceptuel chez l'élève. Table ronde "Didactique des Sciences et Psychologie", Paris, 1977.

[72] HEBENSTREIT, J. Bilan et perspectives d'avenir de l'E.A.O. In [92].

[73] HEBENSTREIT, J et LUMBROSO, M. L'ordinateur outil pédagogique de simulation en sciences physiques. In "Le monde international de l'enseignement de l'informatique", vol. 1, n° 2, 1974.

[74] HEBENSTREIT, J. Nouvelles tendances en enseignement assisté par ordinateur. Bulletin de liaison de la recherche en informatique et automatique, n° 39, 1977.

[75] HENNEQUIN, P.L. Programmes de simulation et d'analyse de données avec animation graphique. Université de Clermont-Ferrand.

[76] HENNUY, G. Utilisation conversationnelle de l'ordinateur pour l'entrainement aux applications numériques. Thèse, Université catholique de Louvain, 1979.

[77] HENRY, P. Un système conversationnel de résolution formelle d'équations trigonométriques. Rapport de DEA, non publié, 1976.

[78] HOC, J.M. La programmation informatique comme situation de résolution de problèmes. Thèse, Université René Descartes, Paris, 1978.

[79] HOST, V. Procédures d'apprentissages spontanées - Rapport introductif. Table

ronde "Didactique des sciences et psychologie", Paris, 1977.

[80] HOWE, J.A.M. Developmental stages in learning to program. Research Paper n° 119, Department of artificial intelligence, University of Edinburgh, 1978.

[81] HOWE, J.A.M. et DU BOULAY, B. Microprocessor assisted learning : turning the clock back ? Research paper n° 114, Department of artificial intelligence, University of Edinburgh, 1979.

[82] HOWE, J.A.M. : O'SHEA, T et PLANE, F. Teaching mathematics through LOGO programming : an evaluation study. Research Paper n° 115, Department of artificial intelligence, University of Edinburgh, 1979.

[83] HUC, B. Mise en oeuvre de la méthode de programmation déductive. Thèse, INPL, 1977.

[84] HUTIN, R. Comment concilier un plan d'études par objectifs avec une pédagogie faisant appel à des "situations mathématiques" et à l'apprentissage par conflit. Colloque "Pédagogie par objectifs", Orléans, 1977.

[85] IBM. EPIC : FAST, Student testing for educational institutions.

[86] IMAGO. Louvain la Neuve, Belgique.

[87] INFA et Université de Paris VII. Analyse de quelques textes sur les algorithmes d'enseignement, les méthodes d'organisation didactique de la matière et les analyses de réponses, 1971.

[88] IREM de Nancy. Continuité d'une fonction en un point (cours programmé), 1974.

[89] IREM de Nancy. Dénombrement (cours programmé), 1975.

[90] IREM de Nancy. La loi binômiale (cours programmé), 1974.

[91] IREM de Nancy. Logique (cours programmé, non publié).

[92] IRIA. Compte rendu des journées "Micro-ordinateurs, réseaux et formation". Arc et Senans, 1979.

[93] IRIA. Compte rendu des journées "Compréhension". Arc et Senans, 1977.

[94] IRIA. Compte rendu du séminaire international sur les systèmes intelligents de question-réponse et grandes banques de données. Bonas, 1977.

[95] IRIA. Ecole "Reconnaissance et compréhension du dialogue écrit et parlé". Nancy, 1977.

[96] IRIA. Contributions à l'enseignement assisté par ordinateur. Cahier n° 5, 1971.

[97] JACQUIN R et al. Dépouillement d'une enquête sur les réalisations en enseignement assisté. AFCET (division TTI), groupe de travail "informatique et enseignement", non publié.

[98] JACQUOT, R ; KOLMAYER, E et QUERE, M. EVE. INRP-IREM de Lorraine, 1977.

[99] JAYEZ, J.H. Etude des relations d'organisation de la matière en enseignement assisté par ordinateur. Thèse, Université Paul Sabatier de Toulouse, 1974.

[100] KAYSER, D. Définition d'un critère pour la construction de règles par apprentissage sur ordinateur. Thèse, Université de PARIS VI, 1975.

[101] KAYSER, D. Etude d'un analyseur et d'un compilateur conçus pour l'enseignement assisté. Thèse, Faculté des Sciences de Paris, 1970.

[102] KAYSER, D. Utilisation des connaissances en enseignement assisté par ordinateur. In [93].

[103] KIRCHBERGER, A et al. Quelques aspects de l'enseignement programmé dans le monde. In Enseignement Programmé, Dunod, Paris, 1968.

[104] KNUTH, D.E. The Art of computer programming. Vol. 1 : fundamental Algorithms. Seconde édition. Addison-Wesley, 1973.

[105] KOFFMAN, E.B. et PERRY, J.M. Description and evaluation of a model for generative CAI. In [115], Part 2, pp 889-894.

[106] KOLMAYER, E. et QUERE, M. Activités mathématiques et travail en groupe. Publication de l'IREM de Nancy, 1974.

[107] KOLMAYER, E. et QUERE M. Démontrer des théorèmes : pourquoi et comment ? Publication de l'IREM de Nancy, 1974.

[108] KOLMAYER, E. et QUERE, M. Enseignement Programmé et formation continue des enseignants. Publication de l'IREM de Nancy, 1974.

[109] KOLMAYER, E. Evaluation de la méthode déductive de programmation. Publications du Centre de Recherche en Informatique de Nancy, n° 78-R-041, 1978.

[110] LACOMBE, D. Didactique. Bulletin "Didactique des disciplines", Université de Paris VII, n° 2, janvier 1974.

[111] LAFOND, C. Expérience des 58 lycées. INRP, Paris, France.

[112] de LANDSHEERE, G et V. Définir les objectifs de l'éducation. PUF, Paris, 1975.

[113] LANGTON, N.H ; ELLINGTON, H.I et ADDINALL, E. Science based educational games and simulations. Conseil de l'Europe. Conférence européenne sur le rôle et la valeur des nouvelles techniques de communication dans l'éducation post-secondaire, Strasbourg, 1979.

[114] LANSKY, M. L'informatique éducationnelle. Une approche systématique de l'application de l'ordinateur dans l'éducation. Ecole d'été UCODI, Genève, 1975.

[115] LECARME, O et LEWIS, R. Computers in education. North Holland, 1975.

[116] LECLERCQ, D. Evaluation formative et évaluation sommative sur mesure (par ordinateur) au moyen de banques de questions structurées ou calibrées. Conseil de l'Europe, conférence européenne sur le rôle et la valeur des nouvelles techniques de communication dans l'éducation post-secondaire, Strasbourg, 1979.

[117] LEE, W. et al. Rapport sur l'utilisation de l'ordinateur à des fins pédagogiques. Ministère de l'Education du Québec, 1973.

[118] LEGIS. Karlsruhe, RFA.

[119] LE NY, J.F. La communication entre individus, classes et groupes. In [92].

[120] LE NY, J.F. Le conditionnement. PUF, Paris, 1966.

[121] LE XUAN et CHASSAIN, J.C. Analyse comportementale, Nathan, 1975.

[122] LIVERCY, C. Théorie des programmes. Dunod, 1978.

[123] LOGO. MIT, Cambridge, USA.

[124] LOPATA, G. Bibliothèque de Q.C.M. In [115], Part 2, pp 715-719.

[125] LOVE. IUT de Lyon, France.

[126] LUFT, R. Méthodes d'expérimentation fictive en cinétique chimique par simulation sur ordinateur. Université de Nice, France.

[127] LYMAN, E.R. A descriptive list of PLATO Programs 1960-70. CERL/X-2, University of Illinois, 1970.

[128] MAGISTER et ERASME. Université de Grenoble, France.

[129] MARTON, F. Rapport introductif au symposium "Les stratégies de recherche - développement dans l'enseignement supérieur", Göteborg, 1974.

[130] MARKOV, A.A. Theory of algorithms. Israel Program for Scientific translations, 1962.

[131] de MENDONCA, F. L'utilisation en France de l'E.A.O. In [92].

[132] MEYER, B et BAUDOIN, C. Méthodes de Programmation. Eyrolles, 1978.

[133] MIGNE, J. Les paradoxes de l'évaluation. INFA, non publié, 1970.

[134] MOHR, R. Critique de la présentation des analyses en méthode déductive. Bulletin de liaison du Centre de Recherche en Informatique de Nancy, n° 11, 1979.

[135] MOLES, A.A. Psychologie de la découverte dans le domaine mathématique. Non publié.

[136] Moniteurs SIRIS 7/SIRIS 8 Temps Partagé. Versions B09/C09. Manuel d'utilisation et d'opérations. Documentation CII n° 4503E/FR, 1975.

[137] de MONTMOLLIN, M. L'enseignement programmé. PUF, Paris, 1965.

[138] MOREAU, J.M. Analyses pour un modèle de la transmission d'informations scientifiques. Laboratoire de Psychologie génétique, Paris-Sorbonne, 1970.

[139] MORGANOV, I.B. L'utilisation des graphes dans l'élaboration des programmes. In Enseignement Programmé, n° 1, 1966.

[140] MOUYSSINAT, M. Un code pour la représentation, l'étude et la manipulation de structures de données. Application aux bases de données. Thèse, Université de Bordeaux 1, 1978.

[141] MYCIN. Stanford University, USA.

[142] NDPCAL (National Development Programme in Computer Assisted Learning). Papiers de R. HOOPER et R. MILES. UK.

[143] NEEL, F. Terminal d'aide à la programmation et d'apprentissage de la programmation avec écran, entrée graphique et unité de réponse vocale. LIMSI, CNRS, Orsay, France.

[144] NORA, S et MINC, A. L'informatisation de la société. La documentation française, Paris, 1978.

[145] OPE (ordinateur pour étudiants). Université de Paris VII, France.

[146] PAIR, C. La construction des programmes. RAIRO Informatique, Dunod, vol 13, n° 2, 1979.

[147] PAIR, C. Structure de données et algorithmes fondamentaux. Fascicule II, Cours de 2ème année de l'Ecole Nationale Supérieure de la Métallurgie et de l'Industrie des Mines de Nancy, 1974.

[148] PALANCHINI, A. Analyse automatique de la réponse dans un enseignement assisté de logique. Publications du Centre de Recherche en Informatique de Nancy, 77-R-001, 1977.

[149] PALMER, B.G. et OLDEHOEFT, A.E. The design of an instructional system based on Problem generators. Int. J. Man-Machine Studies, 7, 1975.

[150] PAPERT, S. Teaching children to be mathematicians versus teaching about mathematics. International Journal of mathematical education in science and technology, vol. 3, n° 3, 1972.

[151] PAPERT, S. Uses of technology to enhance education. Report of the M.I.T. n° 298, 1973.

[152] PASK, G. Systèmes d'apprentissage. Gestion de l'élève. Working Paper UCODI n° 8.

[153] PCDP. Université d'Irvine, Californie, USA.

[154] PEELE, H.A. et RISEMAN, E.M. Four faces of HAL : a framework for using Artificial Intelligence Techniques in Computer-Assisted instruction. IEEE Trans. Syst. Man and Cybern., mai 1975.

[155] PERRIAULT, J. Domaines actuels d'utilisation des calculateurs dans l'enseignement. In Enseignement Programmé, Dunod, Paris, 1968.

[156] PERRIAULT, J. Quelques notes sur les problèmes que pose le développement de l'emploi des calculateurs dans l'enseignement en France. In Enseignement Programmé, Dunod, Paris, n° 6, 1969.

[157] PERRIAULT, J. Si l'utilisation des machines à enseigner se développait en France. In Enseignement Programmé, Dunod, n° 7, 1969.

[158] PERRIAULT, J. Un système de représentation des contenus à enseigner avec l'assistance d'un ordinateur. Applications à l'hématologie et à l'économie. OFRATEME, Paris, 1976.

[159] PICO. Université Paul Sabatier, Toulouse, France.

[160] PISTER, C. Contribution à l'étude de l'utilisation d'une base de données dans un environnement d'enseignement assisté par ordinateur. Rapport de stage de DESS, 1979, non publié.

[161] PLAN (Program for Learning in Accordance with Needs). USA.

[162] PLANNER. MIT, Cambridge, USA.

[163] PLATO. University of Illinois, USA et Control Data Corporation.

[164] POINTEAU, C. Analyse canonique. Note interne IRIA, 1970.

[165] POLY, A. Système de correction personnalisée de QCM par ordinateur. ENS de Saint-Cloud.

[166] PRESS, L. Using interactive simulation in teaching linear programming. In [115], Part 2, pp 543-546.

[167] QUERE, A. Etude des ramifications et des bilangages. Thèse, Faculté des Sciences de Nancy, 1969.

[168] QUERE, M. Démarches algorithmique et déductive en enseignement assisté. Rapports avec l'intelligence artificielle. Colloque "analyse de la didactique des mathématiques", Bordeaux, 1975.

[169] QUERE, M. Modèle mathématique d'un système d'enseignement. In [115], Part 2, pp 895-898.

[170] QUERE, M. Propositions pour l'analyse d'un contenu à enseigner en vue de sa programmation et applications à l'enseignement des mathématiques assisté par ordinateur. Publications du Centre de Recherche en Informatique de Nancy, n° 76-R-016, 1976.

[171] QUERE, M. Utilisation des ramifications dans l'analyse d'un contenu à enseigner. In Mathématiques et sciences humaines, n° 53, 1976.

[172] REGNIER, J et DE MONTMOLLIN, M. Reconnaissance de l'organisation, recherche de l'ordonnancement des éléments et choix du mode d'enseignement de la matière. In La Recherche en enseignement programmé, tendances actuelles, Paris, Dunod, 1969.

[173] REINBOLD, M.F. Dossier introductif à l'enseignement assisté par ordinateur. Institut de formation du Crédit Agricole et Mutuel, 1978.

[174] ROMBOUTS, J. Rapport introductif au colloque européen sur l'enseignement assisté par ordinateur de l'institut européen pour la formation professionnelle, Monchy Saint-Eloi, 1979.

[175] RORPACH, M. Langages et systèmes d'implémentation de cours en enseignement assisté par ordinateur. Publications de l'IREM de Nancy, 1974.

[176] ROUANET, H. Les modèles stochastiques d'apprentissage. Mathématiques et sciences humaines, Paris, 1964.

[177] SABAH, G. Contribution à la compréhension effective d'un récit. Thèse, Université de Paris VI, 1978.

[178] SABONNADIERE, J.C. et MASSE, P. Les graphes : un support conceptuel puissant dans la construction de systèmes de conception assistée par ordinateur. Ecole d'été UCODI, Genève, 1975.

[179] SAYETTAT, C. Contribution à la conception, l'élaboration et l'utilisation du système de bases de données SOMINE : aide à la C.A.O. Thèse, Ecole Nationale Supérieure des Mines de Saint-Etienne, 1976.

[180] SCHMITT, A. Simulative transfer of mathematical concepts by interactive programming. In [115], Part 2, pp 811-815.

[181] SCHOLAR. MIT, Cambridge, USA.

[182] SCHOLL, P.C. Vers une Programmation systématique : étude de quelques méthodes, techniques et outils. Thèse, Université de Grenoble, 1979.

[183] SCHRÖTER, G. Transformation des égalités. Armand Colin, 1969.

[184] SIAM-DOCEO II - LPC, Université de Liège, Belgique.

[185] SKEMP, R.R. La psychologie de l'apprentissage et de l'enseignement de la mathématique. Budapest, 1962.

[186] SLEEMAN, D.H. A Problem-solving monitor for a deductive Reasoning Task. Int. J. Man-Machine studies, 7, 1975.

[187] SMITH, R.L. ; GRAVES, W.H. ; BLAINE, L.H. et MARINOV, V.G. Computer-assisted axiomatic mathematics : informal rigor. In [115], Part 2, pp 803-809.

[188] SOCRATE II. Manuel d'utilisation. Documentation CII-HB n° 00F2 4742 REVO, 1977.

[189] SOCRATE II sous SIRIS 7/SIRIS 8. Manuel d'opérations. Documentation CII-HB n° 58 F2 7207 REVO, 1977.

[190] SOPHIE. University of California, USA.

[191] SUPPES, P. Institute for Mathematical Studies in the social Sciences. Stanford University, USA.

[192] SUPPES, P. The computer teaches arithmetic. In school Review, University of Chicago, vol 79, n° 2, 1971.

[193] TALIZINA, N ; BONDINE, O ; LANDA, L ; GOLIKOV, A et GUERASSIMOV, V. Colloque franco-soviétique, 1973. In Bulletin de psychologie, n° 315, 1974.

[194] TAUBER, M. Gestion par ordinateur de l'enseignement à distance. Conseil de l'Europe, Conférence européenne sur le rôle et la valeur des nouvelles techniques de communication dans l'éducation post-secondaire, Strasbourg, 1979.

[195] THALES, Eidgenössische Technische Hochschule Zürich, Suisse.

[196] TICCIT, MITRE Corporation, Virginia, et université Brigham Young, Utah, USA.

[197] TOURNEUR, Y. Classification NLSMA. Mathematica et Paedagogia, n° 56 et 57, 1972.

[198] UCODI. Panel discussion on formalization in teaching and learning activities (using computers). Summer School, GENEVE, 1975.

[199] VAQUERO, A. Analyse du procès d'information dans l'enseignement programmé et application aux machines à enseigner. Ecole d'été d'informatique de l'AFCET, Granada, 1973.

[200] VASKEVITCH, D. Teaching propositional calculus with CAI using APL. In comptes rendus du colloque canadien sur la technologie pédagogique, NRCC n° 12726, Calgary, 1972.

[201] VERMERSCH, P. Analyse de la tâche et fonctionnement cognitif dans la programmation de l'enseignement. A paraître au bulletin de psychologie.

[202] VERMERSCH, P. Comment analyser la conduite de l'élève adolescent ou adulte sous l'éclairage de la théorie de l'intelligence de J. Piaget. Compte rendu d'un séminaire tenu en 1977 au CIEP de Sèvres. Publications du CNTE, Vanves, 1979.

[203] VERRECK, W.A. Rapport introductif au symposium "Les stratégies de recherche-développement dans l'enseignement supérieur", Göteborg, 1974.

[204] VIENNOT, L. Le raisonnement spontané en dynamique élémentaire. Thèse, Université de Paris VII, 1977.

[205] WARNIER, J.D. Les procédures de traitement et leurs données (LCP). Paris, Editions d'Organisation, 1975.

[206] WATANABE, S. Apprentissage créatif et automate de propension. Working paper UCODI n° 9.

[207] WEISBUCH, G. Utilisation des méthodes de simulation sur ordinateur dans l'enseignement des disciplines scientifiques. Université d'Aix-Marseille II, France.

[208] WELTNER, K. Définition de l'organisation optimale de la matière à enseigner. Working paper UCODI n° 2.

[209] WEXLER, J.D. Information Networks in generative Computer-Assisted Instruction. IEEE Trans. on Man-Machine Systems, vol MMS 11, 1970.

[210] WIRTH, N. Algorithms + Data Structures = programs. Prentice Hall, 1976.

[211] WIRTH, N. Introduction à la Programmation systématique. Masson, Paris, 1977.

[212] WOLLMER, R.D. A markov Decision Model for Computer-Aided Instruction. Mathematical Biosciences, 30, 1976.

[213] ZAPPATA, M. La méthode d'interaction constante des unités programmées. Colloque franco-soviétique, 1975. In Bulletin de psychologie, n° 330, 1976-1977.

NOM DE L'ETUDIANT : Madame QUERE Maryse r TOMMASINI

NATURE DE LA THESE : Doctorat ès Scien's

VU, APPROUVE

et PERMIS D'IMPRIMER

NANCY LE 18 JAN 1980 00484

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

M. BOULANGE