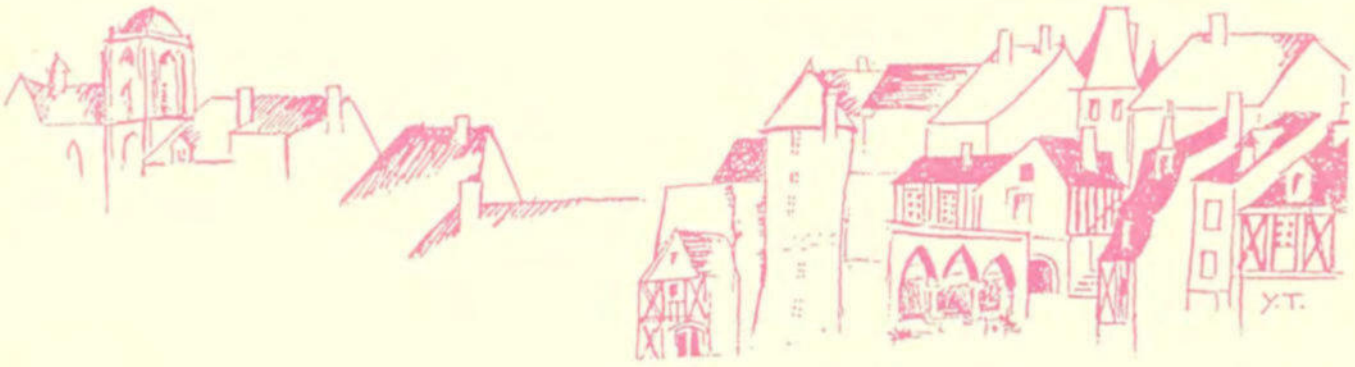
Mathématiques et enseignement intelligemment assisté par ordinateur

Une vision hypertexte des environnements d'apprentissage



THÈSE DE DOCTORAT, Spécialité INFORMATIQUE, présentée par Eric BRUILLARD

LABORATOIRE D' INFORMATIQUE
UNIVERSITÉ DU
MAINE
B.P.535 F72017 LE MANS CEDEX

THÈSE de DOCTORAT de l'UNIVERSITÉ du MAINE

Spécialité :
Informatique

Présentée par :
Eric BRUILLARD

Pour obtenir le titre de :
DOCTEUR DE L'UNIVERSITÉ DU MAINE

Sujet de la thèse :
Mathématiques et EIAO :
une vision hypertexte des
environnements d'apprentissage

Soutenue le 14 février 1991
devant le jury composé de :

Mme	Monique	GRANDBASTIEN	Rapporteur
MM.	Roger	CUPPENS	Examinateur
	Paul	MENGAL	Examinateur
	Gérard	VERGNAUD	Rapporteur
	Martial	VIVET	Directeur
	Harald	WERTZ	Président

L'ensemble des travaux relatés dans cette thèse est le résultat de très nombreuses collaborations qu'il est difficile de toutes mentionner. J'espère que les personnes omises ne m'en voudront pas trop de les avoir oubliées dans l'énumération qui suit. Qu'elles sachent que ce n'est en rien délibéré. L'essentiel reste de pouvoir à nouveau susciter des occasions de mener des travaux fructueux en commun.

Tout d'abord, je remercie Martial Vivet qui a su constamment me soutenir dans ce travail en apportant amicalement toutes les aides nécessaires, malgré un emploi du temps démentiel. Son rôle dans le champ de l'EIAO en France est fondamental et ce travail n'aurait pu être mené à terme sans son concours.

Je remercie Monique Grandbastien d'avoir accepté d'être rapporteur de cette thèse et pour les conseils judicieux qu'elle a pu me donner. Son aide s'est révélée très précieuse pour la rédaction de cette thèse.

Je remercie Gérard Vergnaud d'avoir accepté d'être rapporteur de cette thèse et pour les discussions fructueuses que nous avons eues. Elles m'ont permis de mieux préciser les idées centrales de mes recherches.

Je remercie Roger Cuppens pour les divers conseils qu'il m'a prodigués. Il prouve depuis plusieurs années, dans le cadre de son travail à l'IREM et dans les Universités d'été qu'il organise à Toulouse, son attachement aux recherches sur l'enseignement des mathématiques. De plus, sa rigueur et sa grande culture en font un correcteur exceptionnel.

Je remercie Paul Mengal de m'avoir permis d'abuser de son ouverture d'esprit, de son intelligence et de sa disponibilité. Son regard étranger, mais pertinent et profond sur le domaine a permis d'éclaircir certains points délicats.

Je remercie Harald Wertz qui a accepté de présider le jury. Ses travaux d'une grande qualité scientifique et son intérêt soutenu pour la formation en font une référence. Sa profondeur de vue et ses conseils judicieux ont été d'un grand secours.

Je remercie enfin Gérard Weidenfeld qui a guidé mes premiers pas dans la recherche et m'a soutenu et aidé pendant très longtemps.

Je remercie :

- les personnes de l'ex-Centre Mondial de l'Informatique, et plus particulièrement Y.D. Pérolat, N. Rocland et F. Chauveau;
- les équipes de l'Institut National de la Recherche Pédagogique (DP5), avec une mention spéciale à Frédéric Robert, ses connaissances sur la grammaire et les engrenages vélocipédiques (Logo), son humour et sa grande probité intellectuelle, ainsi que J.F. Boudinot, J.C. Letouzé, et J. Perriault. De même :
  - F.M. Blondel et toute l'équipe du projet chimie,
  - l'équipe du projet mathématiques : Bernard Dumont, J.F. Boudinot, J.P. Drouhard, B. Grugeon, Y. Paquelier et C. Terlon,
  - N. Salamé;
- Monique Baron, J. Mathieu, E. Cauzinille-Marmèche;
- les personnes de Softia : G. Weidenfeld, E. Ferret, et Paolo Machado qui n'a pas

ménagé ses efforts pour fournir des outils performants pour le développement des applications éducatives;
- les auteurs J.B. Melet, Marina Médiavilla, J. d'Yvoire;
- les personnes de l'Ecole Normale de Bonneuil et plus particulièrement C. Drouin, F. Boumati et C. Gauthereau;
- l'équipe des thésards du Mans et plus particulièrement Elisabeth Delozanne.

"Last but not least", le document final n'aurait pu voir le jour sans l'énorme travail réalisé par Nicole Rocland, qui a transformé un brouillon ASCII en une merveille Latex. Son expertise et sa gentillesse montrent que les techniques les plus performantes n'existent que par l'intelligence humaine. Enfin, je remercie la société C2V qui a fourni gracieusement les moyens matériels nécessaires au tirage du document de base.

Première Partie

Introduction

« Imaginons, dessiné dans un espace de représentation, un diagramme en réseau. Il est formé, pour un instant donné (car nous verrons amplement qu'il représente un état quelconque dans une situation mobile), d'une pluralité de points (sommets) reliés entre eux par une pluralité de ramifications (chemins). Chaque point représente soit une thèse, soit un élément effectivement définissable d'un ensemble empirique déterminé. Chaque chemin est représentatif d'une liaison ou rapport entre deux ou plusieurs éléments de cette situation empirique...
Il est possible de découper sur la totalité du réseau des sous-ensembles restreints, localement bien organisés, tels que leurs éléments soient plus naturellement référables à cette partie qu'à l'ensemble total (bien qu'en droit ils soient toujours référables à lui). En s'organisant par parties, ces éléments forment une famille à puissance déterminante locale plus forte que si l'on additionnait purement et simplement leur puissance respective de détermination...
Il (ce réseau) brise définitivement la linéarité des concepts traditionnels : la complexité n'est plus un obstacle à la connaissance ou, pis, un jugement descriptif, elle est le meilleur des adjuvants du savoir et de l'expérience. »

Michel SERRES, Janvier 64
"Le réseau de communication : Pénélope"
[SERRES 68]

"Let the student pick what he wishes to study next, decide when he wishes to be tested, and give him a variety of interesting materials, events, and opportunities... Under such circumstances, students will actually be interested, motivated to achieve far more than they have ever achieved within the normal instructional framework... If they start soon enough they may even reach adulthood with natural minds : driven with enthusiasm and interest, crippled in no areas, eager to learn more, and far smarter than people ordinarily end up being. "

Nelson, 1970 [NELSON 70]
cité dans [SHNEIDERMAN & KEARSLEY 89]

I.1.a Prolégomènes

Ce travail se situe dans le domaine de l'Enseignement Intelligemment Assisté par Ordinateur, essentiellement dans le cadre de l'apprentissage des mathématiques. Il est totalement imprégné d'une vision hypertexte, tant au plan des réalisations décrites, qu'au plan de l'exposition elle-même, puisque la rédaction est réalisée en hypertexte.

Dans ce contexte, une introduction classique, développant une argumentation sur un mode linéaire, est impossible. L'introduction se présente ici comme une suite de thèmes, à la fois autonomes et fortement inter-connectés : des points de vue complémentaires et convergents.

La citation précédente de Michel Serres éclaire cette tentative. Sa description du « réseau de communication Pénélope » est directement inspirée de son travail sur Leibniz. Leibniz, qui fut créatif aussi bien dans le domaine scientifique que dans le domaine philosophique, peut d'ailleurs être considéré comme l'un des précurseurs de cette approche. La forme choisie cristallise un mode d'approche du domaine, une 'expression hypertexte' de l'interdisciplinarité dont on peut trouver les premiers échos dans les travaux de Leibniz au travers de l'éclairage de Michel Serres. Cette filiation entre les techniques de l'ère informatique et les réflexions profondes d'un des grands penseurs de la tradition philosophique devrait certainement être analysée plus complètement.

En aucun cas, ce travail ne doit être vu comme un aboutissement, mais comme une étape. C'est un point de départ, l'initialisation d'une forme par essence en devenir.

Note : une lecture linéaire est néanmoins possible, la forme imprimée du document privilégie une circulation parmi les multiples possibles.

I.1.b Thèmes

L'ensemble des thèmes développés peut être classés en plusieurs catégories :

- les thèmes généraux :
  - Thème 1 : Inter-disciplinarité
  - Thème 2 : Problématiques
  - Thème 3 : Méthodologie (travail, expérimentation, exposition)
  - Thème 4 : Hypertexte (communication)
- les thèmes spécifiques :
  - Thème 5 : Connaissance
  - Thème 6 : EIAO (micromondes, hypertextes,...)
  - Thème 7 : Apprentissage (intelligence, éducation,...)
- un méta-thème (Thème 8) : Auto-référence
- le thème central (Thème 9) : Vision hypertexte des environnements d'apprentissage

I.1.b.1 Thème 1 : Inter-Disciplinarité

I.1.b.1.a E.I.A.O. et inter-disciplinarité

Le domaine de l'EIAO (Enseignement Intelligemment Assisté par Ordinateur) se situe au carrefour de divers champs de savoir (informatique, didactique, psychologie, sciences de l'éducation,...). Cette thèse est une tentative pour mettre en évidence les interactions entre ces divers pôles de savoir dans le cadre de l'EIAO. C'est un choix épistémologique de ne pas se placer du point de vue d'une seule discipline, mais d'essayer de se situer à l'intersection de ces différents axes, tout en gardant cependant une vision principale de nature informatique. Il ne s'agit pas de dénier à chaque champ de savoir sa logique propre, mais d'établir des relations multiples et des opportunités de circulation. En ce sens, on ne cherche pas à unifier pour éviter les contradictions (les heurts peuvent être fertiles) mais à faire converger les intérêts de ces divers champs dans un cadre spécifique.

I.1.b.1.b Problématiques et inter-disciplinarité

La problématique générale de l'EIAO est de concevoir et de réaliser des outils informatiques 'avancés' pouvant s'insérer efficacement dans des processus d'apprentissage existants ou à mettre en place. Elle inclut l'EAO dit traditionnel dans ses tentatives de remédier à ses insuffisances. L'idée fondamentale est de chercher à résoudre un problème d'enseignement à l'aide d'outils particuliers, problème qu'aucune des disciplines constitutives ne peut prétendre résoudre isolément. Les problématiques propres à l'EIAO sont à la fois spécifiques et traduisibles dans des problématiques concernant chacune des disciplines qui la constituent. C'est à chaque discipline qu'il incombe de faire exister son point de vue propre à l'intérieur de la problématique commune.

I.1.b.1.c Méthodologie et inter-disciplinarité

Cette vision de l'EIAO comme une forme d'intersection de disciplines implique la constitution d'équipes comprenant des personnes de divers horizons, chacune étant responsable des apports de son champ de savoir. Dans un fonctionnement idéalisé, ce type d'approche est une nécessité. Malheureusement, des arguments de fait, liés aux contraintes de la vie quotidienne, à l'organisation académique et au fonctionnement des institutions rendent les mises en pratique difficiles.

La séparation des champs de savoir a des effets néfastes sur la recherche :

- séparation dans le temps des interventions (voir I.1.b.3, page 10),
- enfermement dans un cadre rigide, la problématique EIAO n'a plus qu'un côté anecdotique, simple support de questions disciplinaires,
- réification négative : les problématiques de recherche deviennent internes à la discipline, se déconnectent des problèmes à résoudre. Les outils logiciels ou conceptuels développés pour clarifier des problèmes se transforment en objets d'étude spécifiques au domaine considéré. Ils ne répondent plus à des nécessités sociales. Par exemple, on risque de privilégier l'outil par rapport au but, i.e. conserver un outil malgré son inadaptation

relative vis-à-vis de certains apprentissages (dérive techniciste), ou de mener des expérimentations sans s'assurer de la pertinence du contenu enseigné et des conditions dans lesquelles il est enseigné.

Il faut d'ailleurs noter qu'un travail essentiellement informatique situé dans le cadre de l'EIAO est soumis à un phénomène quasiment inévitable d'entropie. En effet, si on dépasse le champ restreint de l'informatique, on est amené à des descriptions très ambitieuses basées sur des visions superficielles ou illusoires masquant la complexité du domaine. L'analyse initiale est peu à peu érodée par les obstacles multiples inhérents à de telles entreprises (absence de collaboration avec d'autres disciplines, impossibilité d'intégration dans les structures existantes, méconnaissance des différents acteurs du système). Cela conduit à une restriction du champ d'application réel du système, cette réduction étant souvent implicite, de prétendues évidences masquant les faiblesses inévitables. L'historique de CAMELEON (voir V.3.c) en est un exemple (changement de problématique).

I.1.b.1.d Communication et inter-disciplinarité

Se pose le problème de traduction entre les divers champs, et, tout d'abord, de l'existence même d'une telle traduction :

- le pari d'une non existence renferme les diverses disciplines sur elles-même,
- l'existence d'un langage unique universel semble peu vraisemblable, chacun pouvant se sentir trahi. L'hypothèse adoptée ici est que l'on peut retrouver une unité dans la diversité des approches, des points de rencontre nécessitant des approximations, des 'a peu près' qui n'impliquent pas une absence de rigueur.

A ce titre, la communication de la recherche fait partie intégrante de la recherche elle-même afin de permettre le dialogue entre divers champs, l'hypertexte étant la forme la plus adaptée. Le point de vue argumentatif est d'ailleurs spécifique, il est basé sur la circulation rendue possible qui ouvre de multiples sens.

Un dernier point concerne les aspects de diffusion de la recherche. Cette diffusion est un élément de la recherche elle-même et ne peut en être séparée. Tout d'abord pour des raisons pratiques, les écarts entre les chercheurs et les enseignants sont un facteur de ralentissement, puisque le début est souvent consacré à une formation interne des participants. Mais surtout, dans le domaine de l'enseignement, la recherche a besoin de se diffuser pour progresser et avoir du sens (voir I.1.b.5, page 15)! On peut remarquer d'ailleurs, qu'à l'heure actuelle, l'une des applications les plus réalistes de l'EIAO semble être la formation des maîtres [CUPPENS 89].

I.1.b.2 Thème 2 : Problématiques

I.1.b.2.a Niveaux de problématiques

On distingue différents niveaux de problématique :

- les problématiques locales, elles sont développées dans les parties successives (voir I.1.c, page 23),
- la problématique globale (voir I.1.b.9, page 20),

- la problématique épistémologique, perspective au niveau 'méta' qui est définie dans cette introduction (définition de l'EIAO, voir thème 1) et dans la forme de rédaction choisie (hypertexte, voir thème 4).

Cette problématique, basée sur une approche inter-disciplinaire des problèmes d'apprentissage, ne doit pas cantonner la recherche dans le court terme et et refuser l'abord de questions de recherche fondamentale, mais doit maintenir une cohérence globale et éviter la perte de sens et les visions trop parcellaires et restrictives : les problématiques disciplinaires ne peuvent exister que comme sous-problématiques.

I.1.b.2.b Unification des problématiques

Bien que l'un des buts de la recherche soit de produire des invariants et des concepts généraux, le problème se pose de savoir si les concepts définis dans le cadre de l'EIAO sont facilement unifiables. Plusieurs obstacles rendent difficiles de telles unifications :

- partage d'un même concept entre plusieurs disciplines (par exemple, la métaconnaissance),
- pluralité des points de vue sur un même concept,
- évolution rapide des concepts, fixation malaisée : un concept vivant a-t-il des frontières bien dessinées? On peut à ce propos rappeler l'opinion de Bachelard : *«c'est au moment où un concept change de sens qu'il a le plus de sens »*.

La notion de micromonde est caractéristique de cette difficulté, dans la pluralité de ses acceptions (voir II.1.a, page 29).

La démarche suivie dans ce travail qui est de faire coexister différents points de vue ne tend pas vers les définitions uniques.

I.1.b.3 Thème 3 : Méthodologie

I.1.b.3.a Les approches

On distingue souvent deux approches [DILLENBOURG & Al. 90] :

- une approche orientée vers la théorie dans laquelle les implantations contribuent à augmenter la connaissance dans les disciplines associées à la science cognitive, et principalement l'intelligence artificielle et la psychologie cognitive;
- une approche orientée vers les produits dans laquelle les implantations constituent le but cherché.

Ce dernier type d'approche correspond plus au travail de cette thèse et à la problématique générale suivie. Il faut cependant « transcender ce manichéisme un peu simpliste ».

Tout d'abord, on peut remarquer qu'une approche pragmatique consiste à ne pas inlassablement regretter qu'une machine ne soit pas identique à un être humain et s'attarder sur ses insuffisances pour atteindre un objectif encore peu vraisemblable, mais à s'appuyer sur ses spécificités pour les faire fructifier et les amener à coexister d'une manière optimale dans les structures existantes.

Ensuite, il faut reconnaître que l'on ne peut réfléchir indépendamment des conditions dans lesquelles les techniques utilisées peuvent effectivement être testées sinon on risque de :

- promouvoir des architectures irréalistes,
- augmenter l'écart avec les utilisateurs/prescripteurs,
- oublier des outils efficaces plus standards, mais moins ambitieux,
- empêcher les perspectives incrémentales (améliorations de l'existant),
- détourner de voies de recherche prometteuses.

Enfin, on peut développer une perspective basée sur la généralisation d'approches ponctuelles qui semblent performantes.

I.1.b.3.b Evaluation / expérimentation

Il ne s'agit pas de faire un travail centré uniquement sur la conception, la réalisation et l'expérimentation d'un système informatique, i.e. de parcourir le cycle de vie classique d'un produit. En effet, on peut prendre l'informatique comme une donnée ou un paramètre et essayer d'évaluer son influence. On met alors généralement en place une évaluation sommative pour déterminer si ce produit fini est efficace après qu'il ait été construit. De l'autre côté [LITTMAN & SOLOWAY 88], on peut tenter de faire rencontrer des problèmes avec des solutions et tenter une évaluation formative durant le processus de construction. Le développement de logiciels d'EIAO s'apparentant plutôt encore à une forme d'art, rien ne peut être considéré comme fini, c'est en fait la deuxième perspective qui est suivie : il y a plus d'intérêt dans les guides de développement que dans la détermination de l'efficacité des produits finis (voir I.1.b.6, page 17).

En fait, la dynamique même de l'EIAO et l'évolution extrêmement rapide des techniques rendent souvent les évaluations a posteriori peu pertinentes (dans le cadre EIAO). Leur transférabilité reste d'ailleurs à démontrer. Idéalement, il faudrait pouvoir mener une expérimentation continue, parallèle au développement, permettant les adaptations rapides et supprimant certaines inadéquations aux modes d'usage difficilement prévisibles (c'est le cas, par exemple, du travail qui a été conduit par Rachel Cohen au Centre Mondial de l'Informatique). Ceci n'est possible que dans le cadre d'équipes pluri-disciplinaires.

Dans le présent travail, aucune expérimentation formelle n'a été réalisée. Les différents tests effectués ne prétendent à aucune rigueur scientifique mais fournissent des précisions non négligeables sur certains effets des activités proposées. La plupart des expérimentations se sont faites dans le cadre de la formation des instituteurs, ou dans du suivi dans les écoles. Il s'agit d'expérimentations tout à fait subjectives, réalisées dans un souci de formation (non d'évaluation), elles en respectent les contraintes. Elles ne donnent que des résultats indicatifs d'une portée limitée, mais sont cependant utiles pour confirmer ou infirmer certaines intuitions. En fait, elles permettent plus de juger de l'acceptabilité des produits que leur efficacité.

La problématique d'évaluation est par ailleurs très complexe dans le domaine de l'apprentissage (voir II.3.a, page 67).

I.1.b.3.c Historique personnel

La méthodologie employée correspond à l'histoire personnelle. Un parcours riche dans les structures d'enseignement (quasiment dans tous les niveaux) et en collaboration avec d'autres

structures (Centre Mondial de l'Informatique, Institut National de Recherche Pédagogique, Softia, Université du Mans) a permis de développer une relative expertise liée à cette multiplicité d'expériences (ainsi qu'une certaine habitude du travail inter-disciplinaire).

Le fait d'enseigner les mathématiques et l'informatique dans une Ecole Normale, ou plutôt d'enseigner comment enseigner ces disciplines à des instituteurs ou des futurs instituteurs favorise une approche pragmatique des problèmes. En effet, les visites effectuées dans les classes permettent une évaluation simple et relativement objective de l'influence des formations dispensées dans le cadre du système éducatif. On peut juger rapidement de l'application de ce que l'on a pu proposer. En particulier, l'échec relatif des mathématiques dites modernes, et les incompréhensions et déviations qu'elles ont suscitées amènent à une plus grande prudence pour l'introduction des innovations. En ce qui concerne l'informatique, les difficultés de mise en place sur le terrain du Plan I.P.T. (Informatique Pour Tous) en 1986, ont pu montrer les obstacles culturels à l'introduction des ordinateurs dans le cadre scolaire formel.

La collaboration avec différentes institutions introduit des contraintes que l'on peut juger productives, par exemple :

- avec l'INRP : recherche action, confronter des réalisations à des élèves,
- avec SOFTIA : développement de produits commercialisables,
- cadre Ecole Normale : production informatique utilisable en formation ou dans les écoles à l'issue d'une formation.
- cadre universitaire : développement de problématiques de recherche.

Note : La formation d'enseignants est finalement un travail avec des 'tuteurs intelligents' humains.

I.1.b.4 Thème 4 : Hypertexte

L'un des objets de la thèse est la forme de la thèse elle-même dans une tentative de démonstration pragmatique de l'intérêt d'une conception et d'une lecture hypertextes dans le domaine de l'EIAO. Trouver de nouveaux liens, i.e. structurer des informations disparates, est une forme de nouvelle connaissance (voir I.1.b.5, page 15). Le texte qui suit essaye de motiver le choix d'une telle rédaction et d'en montrer la spécificité.

I.1.b.4.a Du texte linéaire à l'hypertexte

Un texte rédigé d'une manière classique ne se prête pas forcément bien à une forme hypertexte. Il semble d'ailleurs difficile de passer d'une rédaction linéaire à un hypertexte. Alschuler [ALSCHULER 89] tire quelques conclusions de l'analyse d'une publication en format hypertexte de six papiers présentés à la conférence HYPERTEXT '87 avec trois produits différents :

- HyperCard sur MacIntosh (Nicole Yankelovitch IRIS),
- Hyperties sur Pc et compatibles (Ben Shneiderman, University of Maryland),
- KMS sur SUN-3 (Elise Yoder, société KSI). Les résultats sont plutôt décevants et on constate des différences qualitatives dues en particulier à :
- l'adhérence inconsistante à la structure du document,

- la nature subjective des systèmes de liens.

Il ne peut s'agir ni d'un manque de temps ni de talent, vu la qualité des auteurs. Alschuler rapporte que Ben Shneiderman lui a affirmé que son propre travail sur 'Hypertext on Hypertext' avait confirmé ses soupçons sur le fait que ce type de texte 'linéaire' ne pouvait être traduit en hypertexte. Ce relatif échec incline à être prudent dans la production des hypertextes. «*Il est possible que la linéarité du texte imprimé tant raillée dans la littérature hypertexte contraigne moins le lecteur que les boucles aveugles et les liens directionnels d'un hypertexte subjectif* » [ALSCHULER 89].

Cette expérience peut aussi s'interpréter par le fait qu'il est difficile de construire un hypertexte a posteriori, i.e. après la production d'un texte linéaire. De plus, la création des liens est productrice de sens et ne peut être rajoutée simplement sur le texte initial lui-même comme un simple collage, puisque l'hypertexte rend explicite la structure profonde de la connaissance d'un domaine (Raymon et Tompa 87). D'après Carlson [CARLSON 89], bien qu'il n'y ait pas de modèle universel d'implantation d'un hypertexte, l'expérience suggère que, si le document est fortement entrelacé de stratagèmes rhétoriques, la décomposition en unités et liens sera difficile, et de la perte d'informations et des confusions de sens interviendront potentiellement dans le résultat.

Le livre " HyperText and Learning " [JONASSEN & GRABINGER 90], auquel j'ai participé, est aussi accompagné d'une version hypertexte. Cependant, le mode de production du document texte final a intégré les spécificités d'une telle version et favorisé la création collective des auteurs participants. La rédaction s'est ainsi déroulée en plusieurs phases :

- un séminaire où chacun des participants a pu présenter une première version de son papier et le confronter aux autres;
- un premier document a pu être envoyé à chacun avec l'ensemble des textes révisés par les auteurs à la lumière des discussions du séminaire;
- sur cette version, chaque auteur a pu :
  - faire des ultimes corrections sur son propre texte,
  - faire des renvois à d'autres textes du document,
  - annoter d'autres textes du document.

Ce processus a été mené à l'aide d'un réseau, ce qui a permis des itérations sur les différentes références et annotations créées. L'avantage d'un tel mode de production est de permettre aux auteurs eux-mêmes d'élaborer collectivement une version hypertexte et d'éclairer leur discours par référence ou opposition à d'autres discours eux-mêmes pouvant avoir d'autres éclairages. Ceci réduit les risques de vision subjective, puisque la création s'appuie sur les idées d'une communauté.

I.1.b.4.b Pourquoi rédiger cette thèse en hypertexte?

Plusieurs raisons motivent le choix d'une telle rédaction :

- le domaine lui-même : l'EIAO est une discipline carrefour qui n'existe que par les liens qu'elle tisse avec des sciences plus classiques (voir I.1.b.1, page 8). On peut citer d'ailleurs d'autres réalisations du même type comme les nombreux hypertextes sur l'hypertexte lui-même (par exemple 'Hypertext Hands-On! [SHNEIDERMAN & KEARSLEY 89]), ou le travail Intermaps dans le cadre des projets Delta [CLAES 89].

- le type de travail effectué : il est constitué par différentes réalisations, à la fois indépendants et complémentaires, sous-tendues par une perspective commune (voir I.1.b.9, page 20). Les parties sont autonomes et la réflexion surgit des réseaux de liens entre elles, non d'une progression dans l'ensemble du document.
- une nécessité de synthèse : les idées développées ne peuvent être comprises qu'en les situant dans la dynamique du champ très mouvant qu'est l'EIAO (voir I.1.b.2, page 9). Comme le rappelle Michel Serres [SERRES 68] dans son travail sur Leibniz : «*la Découverte n'était pas pour lui liée à une table rase des précurseurs, mais au contraire à une accumulation méthodique de la tradition et de sa réactivation : trouver des veines d'or dans les rochers stériles* ».
- une forme d'auto-référence (voir I.1.b.8, page 19).

Ce choix témoigne aussi d'un parti pris : ne pas séparer la recherche du mode de production et de gestation des idées. Contrairement aux options de J.Pitrat [PITRAT 90], qui cherche à mettre en évidence les niveaux de connaissances différents, ce travail essaye d'intégrer, non de séparer « le livre du méta-livre ». L'aspect informatique dépasse les astuces éditoriales telles celles du livre de Stafford Beer, rapportées par J.Pitrat (4 couleurs différentes : pages blanches pour le texte, pages orange pour les définitions des termes, pages bleues pour structurer le livre, pages jaunes pour expliquer la démarche de l'auteur). La tentative présente devrait assurer une meilleure accessibilité en terme de temps (instantanéité) et d'espace (pas de séparation physique). Certaines astuces de présentation pourront néanmoins reprendre les idées illustrées par le livre de S.Beer.

I.1.b.4.c Contraintes de rédaction

Le travail informatique est décrit en annexe 1 (voir A). La forme imprimée reflète néanmoins les contraints inhérentes à ce type de rédaction. La lecture du document écrit est certainement d'un meilleur confort pour le lecteur sur le plan local et séquentiel, mais dans les parcours à travers les références croisées, il a lui-même la charge d'aller chercher la destination d'un lien.

Le mode de production de cet hypertexte est tout à fait empirique. Dès le départ, la rédaction s'est effectuée en incluant cette perspective, mais aucune méthodologie précise n'a pu être dégagée. En fait, la réalisation a suivi un principe intuitif de convergence. Les différents thèmes ont été soit prévus à l'avance, soit ont émergé durant la phase de rédaction et au cours des relectures successives. L'absence de maîtrise initiale de la globalité du contenu induit une recherche dynamique de liens et d'invariants (incomplétude structurelle, voir thème 8). La nécessité d'autonomie de chacune des parties et de leur lisibilité linéaire ne conduisent pas au découpage fin (dans lequel chaque élément ne contient à la limite qu'une seule idée) qui est une des règles d'or (voir III.1.a.3.d, page 84) de l'hypertexte.

Il est difficile de préjuger de la façon qu'aura le lecteur d'appréhender le texte, mais on peut imaginer quelques points d'achoppement :

- un aspect collage : le point de vue adopté est plutôt analytique dans la mise côte à côte de réflexions, des aspects plus critiques et plus différenciés font peut-être défaut. L'élaboration de synthèses bute sur l'inter-disciplinarité du domaine (voir I.1.b.1, page 8). Un auteur isolé ne peut les prendre en charge complètement, une forme de collaboration est indispensable. De plus, la coexistence locale de différents points de vue est difficile

(unification des problématiques, voir thème 2). La conviction résulte d'un cheminement non réductible à de simples notes, d'une expérience individuelle du lecteur : l'adhésion du lecteur ne s'obtient pas par un discours linéaire progressant pas à pas, mais plutôt par une accumulation d'indices et de points de vue convergents. Il faut enfin remarquer que représenter la complexité répond à deux exigences contradictoires :

  - viser l'utile, le nécessaire, ce qu'il faut pour agir (vision locale),
  - l'encyclopédisme, la prise en compte de multiplicités d'approches (vision globale).

- des possibilités de redites : sont-elles à proscrire? En effet, la pluralité des chemins ne permet pas de savoir ce qui a été lu. Une certaine circularité est ainsi indispensable si on veut éviter une surabondance de renvois. De même, on peut mettre en doute le principe d'identité : une idée reste-t-elle toujours la même chose quand on l'aborde par un chemin différent (voir I.1.b.8, page 19)?
- un côté allusif (voir I.1.b.1, page 8) : l'existence d'allusions correspond au double aspect incomplet et en devenir de cette entreprise :
  - liens non encore explicités,
  - imperfection du document : on ne peut pas tout dire, il reste de nombreuses directions à explorer et de nouveaux éclairages à inclure.

Les thèmes et axes de lecture sont décrits plus loin (voir I.1.c.2, page 23).

I.1.b.5 Thème 5 : Connaissance

I.1.b.5.a Nouvelles formes de connaissances

L'utilisation des ordinateurs amène une révolution en profondeur des modes d'accès à la connaissance et conduit à une révision et à une transformation de ces connaissances elles-mêmes, et de leur transférabilité aussi bien entre des personnes différentes que pour des domaines distincts.

Ce changement d'accès à la connaissance est à la fois :

- quantitatif : accumulation, création de nouvelles connaissances ou modification des priorités dans les hiérarchies classiques,
- qualitatif : introduction de modèles exécutables rendant concrets (ou réifiés) des concepts abstraits. Ceci oblige à expliciter les passages du déclaratif au procédural. L'introduction de modèles graphiques (ou «visuels») permet une forme de transfert du logique à l'analogique, et la prise en compte de styles d'apprentissage différents.

Dans le cadre EIAO, on a une double transposition des savoirs :

- transposition didactique : savoir savant / savoir enseigné,
- transfert d'expertise : savoir expert humain / savoir machine.

Ces transpositions ne sont ni successives ni directement combinables, les interactions avec le savoir sont multiples. Elles ne peuvent se réduire :

- ni à la vision didactique, puisque l'usage de l'ordinateur peut introduire des savoirs 'très' savants (e.g. le calcul formel),

- ni à la vision de l'Intelligence Artificielle, puisque ce n'est pas la vision de l'expert qui est nécessairement requise.

Ceci pose deux problèmes :

- quelle expertise retenir? Est-elle détenue par des individus ? (La réponse me semble être non),
- quelle institutionnalisation donner à ce 'nouveau' savoir?

Ce débat, qui illustre complètement la problématique générale de cette thèse, réintroduit une perspective inter-disciplinaire (voir I.1.b.1, page 8), réductible à aucune des disciplines, sans pouvoir justifier la création d'une nouvelle discipline.

Dans un cadre d'apprentissage, on peut ajouter la distinction entre le déclaratif et le procédural, que l'on peut traduire comme l'opposition savoir théorique / savoir pratique.

Il ne faut pas oublier les déviations du savoir (largement générées par l'enseignement). Les connaissances n'existent pas en dehors des individus, autrement dit, les modèles erronés d'un domaine font partie intégrante de ce domaine. La séparation entre un corpus de connaissances et les représentations des débutants et des experts (les experts pouvant d'ailleurs être efficaces tout en raisonnant avec un modèle incorrect) est arbitraire. L'accumulation et la réorganisation d'un domaine sous forme d'une théorie déductive effacent les écarts à la norme, font oublier les difficultés et négligent les étapes souvent indispensables à la compréhension.

Dans cette nouvelle approche de la connaissance, la notion d'hypertexte joue un rôle central, voire fédérateur (voir I.1.b.9, page 20).

I.1.b.5.b Aspects importants de la connaissance

On peut énumérer différents aspects fondamentaux du traitement de la connaissance dans le cadre EIAO :

- représentation et organisation (forme exécutable ou uniquement consultable),
- modes d'accès et liens entre les connaissances,
- explicitation de la métaconnaissance,
- visualisation, représentations intermédiaires et réification.

Remarque : le terme de 'réification' n'est pas pris ici dans dans son acception marxiste lié au « fétichisme de la marchandise ». Il est utilisé dans un sens positif pour désigner des représentations concrètes, sur lesquelles on peut opérer, de processus de pensée abstraits. L'expression 'réification négative' est cependant introduite pour signifier la déviation consistant à transformer un outil, permettant de résoudre des problèmes, en un objet d'étude en lui-même (forme de glissement de la problématique générale de l'EIAO, voir I.2.a.3).

En mathématiques, les notations constituent un aspect réifié très partiel de la connaissance. Leurs imperfections (ce qu'elles ne montrent pas ou n'induisent pas, et d'un autre côté ce qu'elles risquent d'induire indûment) compliquent les apprentissages. Il en est de même, dans de nombreux domaines, où la forme d'une solution ne reflète pas le processus de recherche qui a permis de la découvrir.

On peut faire une dernière remarque au niveau de la connaissance en liaison avec les problèmes de pédagogie différenciée. En insistant sur le côté social et partageable de la connaissance, sur son acquisition dans des activités de collaboration, on s'éloigne de la prise en compte fine de l'apprenant. L'individualisation, par certains aspects, peut s'opposer à la norme, aux contraintes sociales.

La connaissance n'est pas indéfiniment adaptable à l'individu, sinon elle disparaît (on retrouve un problème similaire dans l'exploration entièrement libre d'un hypertexte, voir III.1.b.1). La question consiste à savoir s'il faut aller vers l'élève ou vers la connaissance.

Ainsi, il faut pouvoir séparer modèle de l'élève et modèle de l'utilisateur :

- le premier doit aider à situer l'apprenant par rapport aux savoirs mis en jeu,
- le second correspond plus à des caractéristiques individuelles liées à des modes d'usage et d'acquisition (modes d'appropriation).

Autant il semble primordial de favoriser la différenciation dans le second cas, autant il semble nécessaire de réfléchir aux limites des éléments individuels dans le cadre du premier. Dans le cadre d'un apprentissage social, un aspect prescriptif ne peut être négligé.

I.1.b.6 Thème 6 : E.I.A.O.

I.1.b.6.a Définition de l'EIAO

L'EIAO peut se définir «syntaxiquement» comme l'intersection de l'IA (Intelligence Artificielle) et de l'EAO (Enseignement Assisté par Ordinateur), i.e. l'ajout de techniques et la prise en compte de concepts venant de la science cognitive (voir I.1.b.1, page 8). Une définition qui me semble préférable consiste à considérer l'EIAO comme l'articulation de tous les thèmes de cette introduction.

I.1.b.6.b Evaluation et EIAO

L'évaluation est un thème récurrent (voir II.3.a). Signalons ici deux problèmes généraux :

- la forme tutorielle facilite l'évaluation puisqu'elle donne le maximum d'initiative à la machine en mettant de côté d'autres formes d'enseignement. En particulier cela permet de mieux situer les interventions extérieures ce qui offre des garanties de reproductibilité. Ce type de travail se pratique plutôt avec des apprenants adultes.
- le double rôle chercheur/enseignant privilégie les études post-bac dans lesquelles le chercheur peut être directement impliqué en tant qu'enseignant. S'ajoute aussi le problème de la double compétence (à la fois en E.I.A.O. et sur le domaine choisi) ou la possibilité d'un travail en équipe. La généralisation pose énormément de problèmes et rend plus difficile l'expérimentation dans les classes secondaires ou primaires.

On trouvera ainsi beaucoup d'exemples sur l'apprentissage de la programmation. C'est pourtant un domaine ne bénéficiant pas d'une grande tradition d'enseignement (et pour cause!), l'utilisation de la machine étant de plus quasiment une obligation.

I.1.b.6.c EIAO et Mathématiques

Le domaine mathématiques a certaines particularités importantes dans le cadre de l'EIAO. En particulier, c'est un domaine fortement formalisé, très proche de l'informatique et qui peut être partiellement automatisé. C'est aussi le lieu privilégié de la résolution de problèmes et de la démonstration. Il faut aussi souligner l'ambiguïté de son rôle dans l'enseignement scolaire obligatoire. Enfin, le problème consistant à introduire un ordinateur pour enseigner les mathématiques ne peut plus se poser en ces termes puisque l'ordinateur est d'ores et déjà indispensable aux mathématiques (même scolaires) elles-même.

I.1.b.7 Thème 7 : Apprentissage

I.1.b.7.a Généralités

Le but de l'EIAO est de permettre de favoriser les apprentissages. A ce titre, il faut poser les problématiques générales de l'apprentissage dans ce cadre particulier :

- **l'intelligence :** elle n'est pas uniquement rationnelle. L'une des critiques formulées sur Logo est le fait que penser ne peut se limiter à créer des procédures [CRAHAY 87]. On pourrait alors caractériser l'intelligence comme une aptitude à créer dynamiquement des liens (vision hypertexte), mais elle ne se réduit sûrement pas à cela. Les deux propositions sont en quelque sorte des facettes distinctes, ayant une part de vérité, leur unification semblant encore difficile (voir I.1.b.2, page 9);
- **la connaissance :** (voir I.1.b.5, page 15) il ne faut pas négliger son aspect social (Vigotsky). Sa représentation est ardue, il n'existe souvent pas de formalisation (elle est incomplète, partiellement fausse, ou trop complexe pour être enseignée). On n'a d'ailleurs pas assez de connaissances, à la fois sur les divers domaines et au niveau 'méta', les difficultés de représentation en attestent, l'enseignement génère en plus ses propres formalisations floues génératrices d'erreur;
- **les processus d'apprentissage :** différentes théories coexistent (voir II.1.c.4.a, page 44), aucune n'est complète. On peut cependant s'accorder sur le fait que tout apprentissage nécessite une durée importante, et que l'on ne contrôle pas l'apprentissage, on peut tout au plus mettre en place des conditions qui le favorisent;
- **les problèmes d'enseignement :**
  - que faut-il apprendre eu égard à l'évolution rapide des connaissances,
  - comment enseigner : l'enseignement humain ne peut être qu'une source d'inspiration, il faut découvrir de nouvelles interactions (apprentissage collaboratif)

Les aspects essentiels sont peut-être dirigés dans le recherche du transfert, les clés de l'apprentissage étant liées à la métacognition, la motivation et l'image de soi.

I.1.b.7.b Informatique et Apprentissage

Il n'y a pas de solution technique aux problèmes d'apprentissage : les problèmes d'éducation sont à la fois complexes et résistants. Ce travail n'a aucune prétention à apporter des solutions réelles, mais se contente uniquement de suggérer de petites améliorations. Il faut abandonner

l'illusion du remède miracle. En particulier, il ne faut pas espérer que l'informatique pourra résoudre les problèmes d'échec scolaire. Elle risque d'ailleurs, au moins dans un premier temps, de profiter plus aux élèves les plus brillants.

On peut aussi noter que la structure même de l'école et les conditions pratiques d'utilisation empêchent la mise en place réelle d'un usage suffisant de l'informatique dans l'éducation (c'est certainement le cas de Logo, voir II.3.a.3).

Inclure un ordinateur pour résoudre un problème d'enseignement change le problème lui-même : il est ainsi délicat de trouver des situations de référence. De la même manière, l'apport de nouvelles techniques s'effectue en réponse à des problèmes ou à des questions et contribue à créer ou modifier les situations existantes. Il est difficile d'isoler des paramètres pour conduire des évaluations, qui nécessitent un temps long.

Avec les nouvelles technologies, au delà des apprenants, les personnes concernées sont des utilisateurs ou des usagers. Se pose ainsi le problème fondamental des modes d'usage, qui englobe les problèmes de formation.

I.1.b.8 Thème 8 : Auto-référence

«*Le système leibnizien s'auto-explique en s'appliquant indéfiniment sur soi-même* » [SERRES 68]

I.1.b.8.a Auto-référence et EIAO

La notion d'auto-référence est une sorte de méta-thème articulé autour des autres thèmes. Elle est au coeur de ce travail :

- dans la rédaction hypertexte (voir I.1.b.4, page 12),
- dans la notion de réification (voir I.1.b.5, page 15), idée générale de la forme reflétant le fond.

On peut s'interroger d'ailleurs sur le sens que l'on pourrait donner à chacun des thèmes appliqué à lui-même (exemple, apprendre à apprendre, hypertexte sur l'hypertexte, un E.I.A.O. sur l'E.I.A.O., connaissance de la connaissance,...).

Les analyses conduites dans le domaine de l'EIAO s'appliquent sur l'EIAO lui-même. En particulier :

- rôle des connaissances implicites,
- l'aspect 'méta'.

Cette notion d'auto-référence conduit, sur le plan théorique, à :

- l'incomplétude (Gödel, aspect qui serait à creuser),
- l'indéterminisme (voir « Les anamnèses mathématiques » [SERRES 68]).

Voir aussi la mise en cause du principe d'identité (I.1.b.4, page 12).

I.1.b.8.b Auto-référence et modes d'usage

On peut attribuer à l'auto-référence un sens pratique : développer des outils pour les autres, cela peut-être aussi les développer pour soi-même, ou tout au moins, réfléchir à ce que l'on voudrait développer pour soi-même. La croyance dans l'intérêt des nouvelles technologies pour la formation n'est cohérente que si on les intégre personnellement.

Un produit développé pour ses propres besoins et détourné à un usage plus général se révèle souvent plus performant et plus intéressant que des productions ad hoc (du fait même qu'il n'y a pas de méthodologie générale de conception de tels outils).C'est la cas de CABRI [BAULAC 90]. De plus, on est souvent le premier testeur de ses productions ce qui garantit souvent l'intérêt du produit. Les choix ne sont pas tous conscients, mais la recherche de solutions à des problèmes vécus permet d'éviter de nombreuses erreurs compliquant les modes d'usage. Les produits destinés aux développeurs intègrent souvent des fonctionnalités très particulières que l'on ne comprend qu'à l'usage. Un cas type est celui de la touche d'inversion de deux lettres contigues (dans l'éditeur EMACS) qui permet de corriger une erreur de frappe classique (les premiers relecteurs de ce document en perçoivent certainement l'extrême utilité!).

Les informaticiens ne sont pas toujours les meilleurs utilisateurs de la technologie qu'ils contribuent à développer. On peut y voir deux raisons :

- une forme d'auto-censure : sauvegarder la communication académique avec les autres disciplines en respectant les usages établis,
- l'absence de logique d'usage : se limiter à une compréhension intellectuelle d'une fonction et se désintéresser de ses applications culturelles considérées hors de leur domaine strict de recherche.

L'hypertexte ne fait pas exception à cette règle.

I.1.b.9 Thème 9 : Vision hypertexte des environnements d'apprentissage

I.1.b.9.a Thème central

> **L'idée essentielle de cette thèse consiste à redéfinir l'architecture des logiciels d'EIAO sur la base d'une représentation cohérente des connaissances associant hypertexte et résolveur, une interface s'appuyant sur une réification efficace de cette connaissance dans un mode utilisateur de forme collaborateur favorisant l'obtention d'une production.**

Tout d'abord, l'hypertexte, associé à des techniques désormais classiques de gestion de l'information (bases de données, Intelligence Artificielle), peut être considéré comme la forme de représentation et d'accès à la connaissance (voir II.3.b).

Ensuite, le résolveur correspond à ce que la machine est capable de résoudre et conditionne les environnements que l'on peut construire. Il intervient de manière déterminante dans les quatre modules de l'architecture classique des tuteurs intelligents (voir II.1.c.1) :

- le domaine, comme module expert,
- l'interface, dans les aspects de réification, voire de simulation,

- l'élève, modélisation par perturbations, ou modèle cognitif exécutable,
- l'enseignant, comme semi-résolveur ou résolveur partiel, et dans l'organisation générale des connaissances (curriculum).

Enfin, dans la relation à l'apprenant, une forme exploratoire localement guidée, outil traitant et partageant les connaissances avec l'utilisateur, semble largement préférable à des formes tutorielles imitant le modèle humain traditionnel. Cette exploration devrait s'inscrire dans un travail de collaboration ou de partenariat avec la machine et déboucher sur une production effective. Cette production, i.e. la trace ou le résultat de l'activité élève est intéressante :

- pour lui-même,
- pour l'enseignant,
- pour la classe (travail coopératif entre les élèves).

Cette production, par le résultat final ou par son processus de réalisation conduit aussi à une réflexion sur sa propre pensée :

- en tant que procédé de réification,
- dans les micromondes, en tant que médiateur entre les objets intermédiaires connus intuitivement et les objets abstraits.

Pour résumer, cette thèse défend l'idée de l'importance prépondérante des relations entre le résolveur et l'environnement proposé qui se concrétise dans la réification, unification entre les idées micromonde, hypertexte et apprentissage.

Cette idée rejoint la problématique de Papert [PAPERT 87] qui considère la notion de micromonde comme le concept le plus riche avec lequel travailler et décrit la création d'un curriculum comme la création d'un réseau de micromondes, chacun d'entre eux se centrant sur des régions différentes de la connaissance. On y inclut la notion d'hypertexte comme une forme nouvelle de micromonde, qui est, par certains aspects, complémentaire des micromondes centrés sur le développement de la pensée procédurale.

Le thème général est celui de la représentation des connaissances.

Le tableau suivant essaye de résumer les différents problèmes liés aux connaissances intégrées à un environnement d'apprentissage. On distingue des représentations exécutables (avec un interpréteur) et non exécutables mais accessibles. Elles ne sont pas exclusives, mais se complètent : à une classe de problèmes peuvent être associés par exemple un résolveur et des accès à un hypertexte.

Ce tableau montre les contraintes techniques sur un résolveur (représentation exécutable) et un hypertexte (représentation non exécutable) suivant différents critères :

- la transparence, i.e., ce que l'utilisateur peut connaître du cheminement suivi par la machine,
- la prise en compte des erreurs,
- la visualisation, comment la forme externe reflète les connaissances du système,
- les niveaux de création, de la modification, à l'ajout (incrémental) jusqu'à la création complète,
- la complétude (voir IV.2.d, page 132).

Sur ce dernier point, il est encore difficile d'associer des niveaux de complétude à un hypertexte. C'est certainement une voie de recherche fondamentale : savoir si une information que l'on cherche est présente ou non, et comment y accéder.

	Exécutable	Non exécutable (accès hypertexte)
Transparence	Boîte noire Explication de la trace Explication des choix Explication de la recherche	chemin chemin + contexte chemin + contexte + règles
Erreurs misconceptions	Perturbable trace Perturbable recherche	accès sur erreurs accès sur erreurs + règles
Visualisation	réification représentations intermédiaires	points de vue browser visualisation des liens
Niveaux de création	Procédures (micromondes) Règles (Systèmes experts) Méta-règles	générateur hypertexte déclaration de contextes création de liens
Complétude	complets quasi-complets incomplets structurellement incomplets	?

I.1.b.9.b Autres thèmes

Cette thèse est centrée sur le domaine mathématiques, mais des incursions dans d'autres disciplines (e.g. le français, voir Lyre III.2.b.3) permettent une distanciation par rapport au contenu et un travail réellement inter-disciplinaire.

D'autres thèmes sont développés dans ce document. On peut citer :

- une réflexion sur la conception des systèmes auteurs dédiés (voir IV.3.c, page 140),
- une étude sur le diagnostic automatique (voir **??**).

Enfin, deux thèmes généraux : le transfert et la réification.

1. Le transfert

   Ce problème du transfert est abordé à différents endroits du document. Il intervient en particulier dans l'opposition novice/expert, et l'opportunité de l'enseignement des heuristiques (voir IV.3.e et VI.3.b).

   L'une des idées, cohérente avec le thème central qui vient d'être développé, est qu'il ne s'agit pas de collecter les méthodes de résolution à associer à des contextes, mais de comprendre l'organisation générale des connaissances, sous une forme déclarative pour permettre aux élèves d'engendrer leurs propres méthodes (articulation déclaratif/procédural). Cela induit un travail sur la métacognition, ou plus exactement sur le développement de conduites métacognitives (voir VI.3.a, page 193).

2. La réification

L'idée générale est que la dynamique de la forme, dans l'aspect local, permet d'accéder au fond et à l'aspect global, que la connaissance soit exécutable ou uniquement consultable.

I.1.c Modes de lecture

I.1.c.1 par chapitres

Ce travail se compose de cinq parties qui peuvent se lire comme différents aspects dans la conception, la réalisation et l'usage d'environnements informatiques destinés à l'apprentissage :

Partie 2 : présentation générale du domaine de l'EIAO (micromondes, architecture des tuteurs intelligents), et étude des spécificités des mathématiques. Divers micromondes sont brièvement décrits. La problématique de l'évaluation est aussi abordée.

Partie 3 : hypertexte et enseignement, explicitation de la notion essentielle de point de vue. Des exemples de réalisations sont détaillés.

Partie 4 : étude d'ARRIA, logiciel destiné à aider les élèves à rédiger des démonstrations mathématiques, basé sur la séparation entre la recherche d'une preuve effective et sa rédaction. Montrer comment créer un système auteur spécialisé basé sur les interactions et un semi-résolveur ('shell' transversal sur le raisonnement).

Partie 5 : montrer comment créer de manière réaliste des systèmes de diagnostic et leur rôle dans l'enseignement : plutôt un outil didactique destiné aux enseignants (feedback sur l'enseignement), ou conception d'outils de guidage locaux (intéressant pour des tâches bureautiques ou à partir de résolveurs existants comme dans le domaine des langages de programmation).

Partie 6 : décrire une architecture reprenant les idées initiales de CAMELIA pour concevoir un système capable de visualiser sa recherche d'une solution. On essaye de voir comment intégrer la notion d'évidence à un résolveur mathématique et comment structurer des connaissances mathématiques élémentaires.

Les apports locaux sont dégagés dans chacune des parties. Elles sont néanmoins sous-tendues par une même vision globale, sortes d'instanciations distinctes des mêmes idées (voir I.1.b.9, page 20). La partie 5 consacrée au diagnostic apporte cependant un contre-point à l'ensemble. C'est une incursion dans les problèmes de modélisation et offre encore peu d'intérêt dans un cadre productif. Les annexes associés à chacune des parties approfondissent l'architecture et le fonctionnement de certaines des réalisations informatiques.

I.1.c.2 Visites guidées

La perspective hypertexte amène à intégrer des visites guidées, i.e. le regroupement de thèmes généraux qui se retrouvent avec des éclairages différents dans les chapitres successifs de la thèse. Les idées développées dans ce travail sont voisines de celles exprimées dans le Manifeste de Genève sur les environnements d'apprentissage intelligents [DILLENBOURG & Al. 90]. Ce manifeste donne une vision structurante de l'état de l'art dans le domaine de l'EIAO et peut constituer une grille d'accès à divers thèmes abordés dans cette thèse :

1. ***La vision rationaliste du raisonnement humain est inadéquate pour la conception des environnements d'apprentissage.***

   (Voir les critiques vis-à-vis de Logo et de la conception consistant à limiter l'intelligence au raisonnement procédural, thème 7)

2. ***L'acquisition des connaissances est dépendante du contexte. Le transfert doit être appris.***

   (Voir transfert)

3. ***Métacognition et réflexion ('reflection') sont les clés d'accès à la connaissance.***

   (Voir VI.3)

4. ***Des expériences d'apprentissage significatives ont besoin de temps. Les environnements d'apprentissage doivent refléter les étapes ('stages') de développement.***

   (Voir I.1.b.7, page 18)

5. ***La principale qualité d'un environnement d'apprentissage est d'aider l'élève à s'adapter à de nouvelles situations.***

   (A relier aux problèmes de transfert et de métacognition)

6. ***La quantité de guidage nécessaire à un apprenant n'est pas une constante éducative universelle mais un paramètre à modifier à travers les actions éducatives.***

   (Une perspective réaliste confie à l'enseignant cette tâche difficile, voir guidage)

7. ***L'apprentissage se produit au travers de l'interface (" Good design is good cognitive theory ").***

   (Voir représentations, réification, thème 5)

8. ***La multiplicité est l'âme des environnements d'apprentissage.***

   (voir I.1.b.9, page 20)

Cette première perspective dégage déjà divers thèmes transversaux :

- représentations intermédiaires et réification,
- guidage,
- transfert (auto-debugging , auto-explication).

D'autres thèmes transversaux peuvent être dégagés et feront l'objet d'itinéraires particuliers :

- complétude,
- conception des résolveurs,
- collaboration :
  - élèves entre eux,
  - élèves / enseignants (problèmes de responsabilité, évaluation),
  - élèves / machines,
  - quelles implications?
- la dichotomie local / global (outils de navigation hypertexte, gestion de curriculum, graphes de démonstration, articulation IA preuves, zoom, métacognition),

- l'opposition forme / fond,
- les modes d'usage et l'image qu'ont les utilisateurs des systèmes informatiques.

Remarque : l'intégration de ces différents thèmes dans des chemins sera effective dans la création de l'hypertexte associé à ce document.

Deuxième Partie

Problématique générale : environnements d'apprentissage et tuteurs intelligents

" The concept of 'intelligent learning environment' generalizes the various kinds of systems developed by researchers in AI and Education. From Intelligent Tutoring System (ITS) to Logo-like micro-worlds, these systems vary along a continuum. Their position is defined by the degree of student initiative, their representation of the learner's knowledge, their knowledge of the tutored domain, etc. The combination of 'intelligent' (taken from ITS) and 'environment' (taken from micro-worlds) points out the current shift in research towards contracting this continuum into a union point. "

[DILLENBOURG & Al. 90]

" With the help of good teachers, AI languages and environments can already provide powerful teaching tools "

[Du BOULAY & SLOMAN 88]

Chapitre II.1

Tuteurs et environnements d'apprentissage

II.1.a Bref historique

L'histoire de l'utilisation des ordinateurs à des fins d'enseignement est déjà assez ancienne (elle remonte aux années 50). On peut brièvement rappeler quelques faits marquants (voir [O'SHEA & SELF 83], [SLEEMAN & BROWN 82] ou [WENGER 87]) :

- les programmes linéaires [SKINNER 1968] basés sur le conditionnement opérant et le renforcement positif (psychologie comportementale);
- les programmes avec branchement [CROWDER 59] où l'analyse de la réponse de l'apprenant est prise en compte et détermine les choix du tuteur. C'est le modèle dominant des langages auteurs d'EAO (Enseignement Assisté par Ordinateur) classique (voir IV.3.c, page 140) ;
- les programmes 'génératifs' capables de générer problèmes, solutions et diagnostic associés (en arithmétique par exemple [SUPPES 79]). Ce sont en quelque sorte les ancêtres des programmes d'EIAO (Enseignement Intelligemment Assisté par Ordinateur) actuels, puisque ces 'drill and practice' pouvaient choisir des problèmes d'une difficulté adaptée à la performance globale de l'élève et comparer la solution trouvée à leur propre solution.

Les espoirs suscités par ces nouveaux dispensateurs de savoirs ont favorisé l'émergence de projets d'envergure. On peut citer :

- TICCIT [MITRE 76] (Time-shared Interactive Computer Controlled Information Television),
- PLATO [ALPERT 75] (Programmed Logic for Automatic Teaching Operation). pour les USA, le projet DIANE en France.

On peut mentionner d'autres catégories de programmes à visées éducatives : les simulations, les jeux et les programmes de dialogue (SCHOLAR [CARBONNEL 70] considéré comme un des précurseurs des tuteurs intelligents).

Les insuffisances de l'EAO classique (incapacité à résoudre les problèmes posés à l'apprenant et à maîtriser les connaissances du domaine, manque d'adaptation à l'utilisateur, lourdeur de développement, etc.), les apports de l'IA et de la science cognitive ont conduit à rejeter l'idée qu'une machine de type répétiteur pouvait être un bon dispensateur d'enseignement. John Self [SELF 85a] fait une étude 'très' critique des logiciels scolaires de ce type.

La position actuelle est d'essayer de construire des programmes intelligents (au sens de l'IA) qui s'inscrivent entre deux pôles extrêmes :

- 'Computer as a teacher', i. e. les tuteurs intelligents (ou ITS Intelligent Tutoring Systems) dans lesquels la machine exerce pleinement le rôle d'enseignant,
- 'Computer as a tool', i. e. les environnements d'apprentissage et les micromondes, où la machine est avant tout un outil utilisé par l'apprenant pour résoudre des problèmes ou effectuer diverses tâches.

II.1.b Environnements d'apprentissages et micromondes

Le développement de LOGO [PAPERT 80] et l'idée d'utiliser des micromondes pour l'éducation est une application des thèses piagétiennes sur la construction des concepts chez l'enfant durant la phase des opérations concrètes. C'est l'enfant lui-même, à travers ses expériences, qui est le moteur de l'apprentissage, constructeur de son propre savoir : l'expérience est le meilleur professeur et, pour apprendre réellement, on ne doit pas donner les réponses à un étudiant, il doit les découvrir lui-même [SCHANK & EDELSON 89]. La genèse de LOGO est décrite dans [LAWLER 87].

II.1.b.1 La notion de micromonde

La notion qui est centrale dans l'approche LOGO est le concept de micromonde. Cette notion est relativement floue, sa définition fluctuant au gré des divers auteurs. Il y a des exemples prototypiques, comme la géométrie tortue ou les 'sprites', mais les définitions formelles sont vagues ou restrictives.

D'après Robert Lawler [LAWLER 87], ce sont des mondes virtuels pour l'action créative. Les objets de ces micromondes ne sont ni des poissons ni des oiseaux; ils ont des propriétés communes à la fois avec les objets formels de la science et les objets plus concrets de l'expérience sensible ('common sense experience').

Papert [PAPERT 87] décrit les objets d'un micromonde comme des objets qui d'une certaine façon sont semblables à ceux avec lesquels on travaille dans le monde réel, et d'une autre façon sont semblables à des objets abstraits. Ce sont des objets transitionnels qui vous aident à manipuler les objets abstraits. Cette aptitude à créer des objets transitionnels nous donne un moyen de combler le trou entre les apprentissages intuitifs et formels. Ainsi, il s'agit d'établir une sorte de représentation intermédiaire, un pont (a 'BRIDGE' [BONAR & CUNNINGHAM 88]) entre le monde réel et le monde abstrait.

Groen [GROEN 84] formule certaines contraintes sur cette représentation : un micromonde est une structure incluant certaines propriétés, les deux plus importantes étant :

1. Une transformation peut être annulée pour retourner à l'état précédent,

2. Il doit exister des applications (dans le sens mathématique du terme) sur d'autres structures qui sont des représentations d'actions concrètes dans le monde réel.

Cette définition limite la notion de micromonde à des univers essentiellement informatiques, la capacité de revenir en arrière n'étant pas toujours simple (la géométrie à la règle et au compas peut être considérée comme un micromonde, la gomme n'étant cependant pas toujours l'outil idéal pour revenir à l'étape précédente). Elle caractérise cependant bien l'aspect opératoire des micromondes (mondes où on agit sur des objets) et leurs contraintes de fidélité et de cohérence.

Certains micromondes sont construits pour faciliter l'exploration d'un domaine, comme les géométries non-standards [ABELSON & diSESSA 80], l'algèbre élémentaire [FEURZEIG 87], la dynamique [TEXIER 88]. Ceci correspond à des activités de modélisation : l'apprenant inspecte la structure du modèle lui-même plutôt que simplement observer ses performances. Il s'agit d'une expérience réelle menée par l'intermédiaire d'un monde artificiel (" *Artificial Worlds and Real Experience* " [DiSESSA 87]).

Dans le rapport sur les technologies de l'information et apprentissages de base, Yves Le Corre [LE CORRE 87] donne une définition assez différente : « *Un micromonde est une représentation sur ordinateur d'un sujet d'étude spécifique, qui permet à l'élève d'acquérir des connaissances factuelles dans un certain domaine, par exemple l'optique géométrique... L'élève teste les hypothèses en modifiant certains paramètres et en observant sur l'écran les effets produits par la simulation sur ordinateur.* » Cette formulation ne correspond pas du tout à celle de la tradition Logo, puisqu'elle limite les possibilités d'intervention de l'apprenant à la simple modification de paramètres, il n'y a pas de possibilité d'action créative. En effet, le coeur de la notion de micromonde repose sur les capacités de conception, il ne s'agit pas d'observer, mais d'agir et de construire.

Finalement, la notion de micromonde se rapproche des travaux sur les représentations intermédiaires désormais classiques dans la littérature sur l'EIAO, à la différence qu'avec ces dernières, un cadre plus contraint limite les possibilités créatives des élèves pour favoriser un contrôle de la machine sur leur activité (voir représentation intermédiaire).

Nous optons donc plutôt pour les premières définitions comme un cadre générique pour la notion de micromonde, en acceptant (par continuité et pour mieux intégrer la tradition EIAO) de relâcher les contraintes liées à l'activité créative en conservant les possibilités d'action. En effet, le dessein de Papert est de changer radicalement le mode d'apprentissage des mathématiques en favorisant les expériences personnelles par l'intermédiaire de micromondes [PAPERT 87], position jusqu'au-boutiste qui peut difficilement coïncider avec l'organisation scolaire actuelle (voir II.3.a.3, page 69).

II.1.b.2 Extension de la notion de micromonde

L'une des caractéristiques des micromondes est de pouvoir piloter un environnement par l'intermédiaire d'un langage de commande, les activités d'apprentissage PAR la programmation correspondent à la même approche. Il s'agit de traduire ses intuitions sous la forme d'un programme.

Il y a d'abord des extensions de langages de programmation traditionnels :

- les extensions de LOGO et plus particulièrement celles à base d'objets ([BORNE 83] [DRESCHER 87]),

- SMALLTALK ([GOLDBERG 79] [BORNE & LEMOYNE 87],[BERGERON 88]),
- PROLOG ([ENNALS 84] [NICHOL 88]).

La redéfinition de nouveaux environnements de programmation, censés être plus abordables pour des utilisateurs débutants :

- BOXER [Di SESSA 85],
- TINKER [LIEBERMAN 87].

Ceux-ci permettent l'introduction de nouvelles métaphores (comme les métaphores spatiales pour BOXER) ou de nouveaux paradigmes de programmation (programmation par l'exemple pour TINKER).

Il faut citer aussi le courant très important de la physique qualitative. De tels micromondes permettent non seulement de simuler les phénomènes d'un domaine mais ils peuvent en plus générer des explications sur le comportement étudié. C'est une branche très prolixe qui compte de nombreuses réalisations : étude des circuits électriques [WHITE & FREDERIKSEN 87], laboratoire de physique [KAMSTEEG & BIERMAN 88], optique géométrique [REIMANN 88], chocs élastiques (système DiBi [STUMPF & Al. 88]), laboratoire électronique (ELAB [BOCKER & Al. 89]), 'Alternate Reality Kit' [SMITH 86], etc. De tels micromondes couvrent l'éventail entre la définition donnée plus haut par Yves Le Corre et la définition de la tradition LOGO.

Les micromondes de complexité croissante ([FISCHER & Al. 78], [FISCHER 88]) sont utilisés pour produire des systèmes de formation professionnelle technique. Les compétences cognitives sont enseignées grâce à un entraînement dans des environnements de plus en plus complexes simulés sur ordinateur. On fait porter la pratique d'emblée sur tous les aspects d'un domaine de compétence, mais en la confrontant à des environnements de complexité croissante (l'exemple détaillé est celui du ski, avec l'invention des skis courts et des fixations de sécurité, ce qui a permis de mettre les débutants beaucoup plus rapidement sur la piste). La notion de micromonde est finalement assez voisine de notre définition, mais transposée pour les besoins de performances de la formation professionnelle.

La notion d'environnement d'apprentissage se rapproche de celle de micromonde. La différence réside dans le fait qu'il s'agit plutôt d'outils quelconques mis à la disposition d'un apprenant pour réaliser une tâche, l'apprentissage étant lié à la manipulation cohérente de ces outils pour atteindre l'objectif fixé. Il n'y a pas de contrainte particulière sur la structure de ces outils. C'est le cas des programmes classiques disponibles sur un ordinateur : traitements de textes, tableurs, bases de données, grapheurs, gestionnaires de plans, etc. Il peut s'agir aussi de programmes plus spécialisés comme des logiciels de calcul formel (voir II.2.a.1, page 49) ou des systèmes experts. Par opposition aux tuteurs, la machine ne contrôle pas l'adéquation entre les outils utilisés et l'objectif poursuivi.

Par extension (ou par une forme de dualité), on emploie aussi le mot micromonde pour désigner le monde conceptuel de référence d'une personne face à un problème dans un domaine [CAUZINILLE-MARMECHE & MATHIEU 88]. Dans la résolution de problèmes d'algèbre, les auteurs distinguent ainsi trois micromondes : arithmétique, algorithmique, syntaxique formel. Cela correspond plus ou moins à la notion de schéma en psychologie cognitive (LAWLER).

Les logiciels d'hypertexte et d'hypermédia (III.1.b, page 85), utilisés en mode auteur, constituent sans aucun doute une nouvelle forme de micromondes très prometteuse. Comme le note

Alan Lesgold [LESGOLD 87] à propos de NOTECARDS (III.1.a.2.b, page 79), ils possèdent les caractéristiques des meilleurs outils autorisant un apprentissage par la découverte, et plus particulièrement les procédés de réification qui permettent de « ... *rendre plus explicite, plus compréhensible et plus manipulable, en tant qu'objet de la pensée, le processus qui consiste à modifier le texte en jouant sur les relations existant entre des réseaux complexes d'idées et un texte linéaire.* »

En fait, ils sont complémentaires des micromondes de type LOGO. Ces derniers développent principalement l'aspect procédural de la pensée, tandis que les hypertextes travaillent essentiellement sur l'information et son organisation, non pas sur la façon d'appliquer telle ou telle connaissance pour résoudre un problème mais sur les nombreux liens que l'on peut tisser entre les diverses connaissances (voir l'introduction). Un autre changement peut être observé sur la façon de 'dialoguer' avec la machine. En effet, le production d'un hypertexte se fait en grande partie en utilisant une interface de manipulation directe (II.1.c.3, page 41), ce qui offre un mode de création beaucoup plus analogique qui peut ouvrir d'autres voies que le mode analytique lié à la programmation Logo.

II.1.b.2.a Rôle de l'erreur et de la représentation dans les micromondes

L'erreur est centrale dans l'apprentissage via les micromondes. En effet, l'utilisateur est amené à confronter ses idées intuitives avec des représentations informatisées. Si ces idées sont fausses, les objets du micromonde ne vont pas avoir le comportement attendu. L'effet de rétroaction est théoriquement immédiat, l'erreur, ou plus exactement sa perception, n'ayant pas besoin d'être mise en évidence par un quelconque tuteur artificiel, puisque l'utilisateur est conscient qu'il n'obtient pas l'effet désiré (on peut se rappeler l'exemple paradigmatique du triangle équilatéral avec la confusion de l'angle interne et de l'angle externe). Ceci conduit l'apprenant à réexaminer sa démarche pour localiser la source de l'erreur et y remédier. Bien sûr, la pratique est un peu moins idyllique, l'erreur étant plutôt assimilée à une faute dans l'enseignement traditionnel (V.1.a.2, page 148), son changement de statut nécessite un comportement non traditionnel de l'enseignant (V.3, page 171).

Un cas très frappant est celui relaté par diSessa [diSESSA 82] qui à l'aide d'un jeu de cibles avec la tortue dynamique ('dynaturtle') montre que la plupart des personnes ont une vision de la dynamique naïve correspondant aux idées d'Aristote, non à celles de Newton. En effet, le monde de l'expérience sensible n'est pas newtonien. J'ai répété de nombreuses fois l'expérience avec des instituteurs à l'école normale de Bonneuil, et j'ai pu observer les mêmes résultats : doutes vis-à-vis de la machine, exploration du micromonde (qui n'est pas très aisé à maîtriser), développement de stratégies de pilotage. L'expérience a toujours été enrichissante.

Un deuxième point très important concerne la notion de représentation. A. diSessa [DiSESSA 87] décrit l'expérience d'un étudiant travaillant sur la géométrie sur un cube et réfléchissant à partir d'un développement plan du cube. Ce n'est pas le formalisme universel habituellement utilisé en mathématiques.

" But gradually, he learned to deal with his representation ever more efficiently, until he saw confidently that he had developed a respectable formalism for dealing with problems of cubical geometry. It is an important property of the cube that it allowed this student to build his own way of thinking about it. My conviction is that building little formalisms like this over an extended period of time can be one of the most profound experiences of mathematics : Mathematics can be made. "

Ceci caractérise bien le comportement d'un expert dans la résolution de problèmes, qui passe beaucoup de temps à se représenter ou à modéliser le problème qui lui est soumis et recherche la solution à partir de cette représentation. Une bonne maîtrise de la représentation est indispensable. Ainsi, un des objectifs centraux dans les micromondes est soit de fournir des représentations suffisamment 'naturelles' ou 'évidentes' pour qu'elles soient opérationnelles, soit de permettre de se les construire et de les explorer pour en acquérir une maîtrise suffisante.

L'un des écueils de cette forme d'apprentissage est le manque de soutien que l'on peut fournir à l'apprenant qui suit des démarches personnelles et peut avoir des difficultés à atteindre son but (ce problème est d'ailleurs analogue à celui de la définition d'outils de navigation dans les hypertextes). Ce rôle est normalement dévolu à l'enseignant dans la pédagogie Logo. Finalement, le problème central dans l'utilisation des micromondes est d'arriver à combiner la résolution de problèmes et la motivation de l'apprentissage par la découverte avec un guidage effectif due à une interaction tutorielle. Cette dernière se rapproche d'ailleurs plutôt d'un système d'aide (II.1.c.2, page 39). Voir par exemple [THOMPSON 87].

L'erreur peut être correctement prise en compte si les objets manipulés dans le micromonde sont bien maîtrisés par l'utilisateur. Dans le cas, où les objets sont étrangers, le repérage de l'erreur devient plus problématique. Ainsi, l'erreur linguistique est plus difficile à repérer sur une production (exemple de LOGOGRAM, II.1.b.3, page 35).

II.1.b.2.b Micromondes pour l'apprentissage de l'informatique

Divers micromondes pour l'apprentissage de l'informatique ont été développés dans le cadre du laboratoire Didacticiels du Centre Mondial de l'Informatique, dirigé par Gérard Weidenfeld : le 'microprocesseur' (Jean-Luc Méheust et Gérard Weidenfeld), l'interpréteur Logo (Nicole Rocland et Eric Bruillard [BRUILLARD & ROCLAND 86], [BRUILLARD & PEREIRA 87]), l'arbre (Isabelle Péreira). Ces environnements exploratoires ont été conçus sur des modèles voisins et sont destinés à l'apprentissage des connaissances informatiques de base. Divers modes d'intervention sont prévus, correspondant à des formes de programmation à différents niveaux :

- découverte des constituants de l'environnement, i. e. la situation telle qu'elle se présente et les primitives permettant de la modifier. Dans le cas de l'interpréteur, diverses boîtes représentent des piles et des buffers, un réseau de communication entre ces boîtes symbolise les transferts possibles entre ces éléments, les primitives permettent d'effectuer ces transferts.
- résolution de problèmes en mode pas-à-pas par activation d'une séquence de commandes. Par exemple, une phrase Logo est proposée par le système ou donnée par l'utilisateur, et ce dernier doit faire fonctionner le dispositif pour l'«exécuter». Il peut contrôler visuellement l'effet des primitives invoquées. Un mode CACHE est également proposé dans lequel l'utilisateur ne connaît pas les éléments de la phrase Logo qu'il manipule (ils sont symbolisés par des carrés) et doit effectuer le travail en se basant uniquement sur les effets des primitives sur la simulation.
- écriture d'un algorithme général de résolution. C'est une étape de généralisation et de programmation, l'utilisateur doit réaliser une procédure permettant l'exécution de n'importe quelle phrase Logo.

A tout moment, des aides variées sont accessibles : un mode 'démonstration', un mode permettant le retour en arrière afin de restaurer une situation antérieure, un mode 'cinéma' permettant de revoir le déroulement des opérations à partir d'un instant donné.

On peut aussi faire un travail spécifique sur les erreurs : introduire des phrases syntaxiquement fausses et observer l'effet sur la simulation. Ceci montre bien à quel moment du cycle de l'interprétation, l'erreur est détectée et quel est le message associé. Cette activité effectuée en mode CACHE 'prouve' qu'il n'y a pas d'analyse sémantique ni de recherche d'intention (croyance naïve de beaucoup de débutants), mais un dysfonctionnement dans un processus totalement automatisé. C'est d'ailleurs cette inintelligence de tels interprètes qui permet de localiser aisément les erreurs et leurs causes (II.1.c.3, page 41).

Il faut bien voir que dans la simulation d'interpréteur Logo, il ne s'agit pas uniquement de réécrire Logo en Logo, comme c'est classique dans ce type de langage (e. g. LISP en LISP [ABELSON & SUSSMAN 85], ou PROLOG en PROLOG [SHAPIRO & STERLING 86]), mais de concrétiser un processus abstrait qui est celui du fonctionnement du véritable interprète du langage LOGO. Cette réification s'appuie sur des objets de transition simples (des boîtes et des fils).

Le modèle sous-tendant l'interpréteur peut lui-même être changé pour intégrer certains fonctionnements particuliers de Logo (comme le traitement des opérateurs infixes ou la primitive SI) ou explorer diverses tolérances (suppression des caractères spéciaux). On a une double forme de complexification : niveau d'intervention sur un modèle déterminé (exploration, pas-à-pas, algorithme), changement du modèle pour intégrer d'autres contraintes techniques. Ce type d'environnement exploratoire se rapproche des micromondes à complexité croissante (II.1.b.2, page 31).

L'utilisation de la simulation dans le cas particulier du traitement des infixes et du parenthésage doit aider les élèves à passer entre les micromondes algorithmiques, arithmétiques et algébriques [CAUZINILLE-MARMECHE & MATHIEU 88] (cela n'a pas pu être testé, malheureusement).

A ma connaissance, la simulation d'interpréteur a principalement été utilisée en formation d'enseignants. Son modèle concret (visualisation de l'interprétation) est souvent une révélation pour les maîtres qui ont un modèle naïf et peu cohérent du fonctionnement de l'interprète Logo.

II.1.b.3 Micromondes pour l'école primaire et la formation des maîtres

Ces travaux s'inscrivent dans une action d'études et de recherches portant sur la conception et l'expérimentation d'environnements informatiques autour de LOGO destinés à susciter une démarche constructive dans l'approche des objets manipulés. Ils ont pour cadre l'INRP (Institut National de Recherches Pédagogiques, direction de programmes n°5).

LOGOGRAM [ROBERT 85] [ROBERT 86] est un environnement de réécriture plutôt destiné à l'apprentissage de la langue et la maîtrise de la grammaire. Il est inspiré des travaux de N. Rowe [ROWE 76] et est assez proche d'autres réalisations (en Logo [SHARPLES 85] et en Prolog [KAHN 84]).

Ce type de micromonde pose différents problèmes. Tout d'abord, du côté des élèves, la non-conformité du résultat obtenu avec celui désiré ou attendu n'est pas toujours décelée ou n'est

pas évidente. Les critères d'acceptabilité sont plutôt lâches. De plus, en raison des choix aléatoires, des transformations ne sont pas toujours visibles. Ensuite, les enseignants ont en général peu de connaissances linguistiques et se désintéressent de la grammaire (les ateliers d'écriture remportent un succès plus affirmé, on a l'équivalent en mathématiques, voir IV.2.c.4).

La reprise du modèle de N. Rowe m'a permis de créer un modèle simple de réécriture destiné à la génération d'énoncés de problèmes de mathématiques avec leur solution (nommé REEC). L'idée est d'approcher la formalisation d'une manière duale : on part du modèle mathématique et on l'enrichit par des réécritures successives pour produire un énoncé (voir word problem). A ma grande surprise, les enseignants demandent invariablement une façon de transformer la génération en un test de connaissances (demander à un élève la production du résultat). Ils ne semblent pas croire à l'efficacité d'une telle approche avec les élèves.

PROD [BRUILLARD 86a] est un environnement destiné à déclarer des règles de production pour la transformation de mots : mise au féminin d'un adjectif, pluriel d'un nom, formation des adverbes, etc. La partie condition porte sur la terminaison d'un mot ou son appartenance à une liste définie, la partie action précise la transformation à effectuer sur les mots. On écrit ainsi des règles du type : si term? "f alors remplace "f "ve (si terminé par "f alors remplace "f par "ve , exemple vif-vive) Au départ, le système n'a aucune connaissance, le but est de pouvoir écrire de telles règles et ajouter des cas particuliers (on précise alors le mot et son transformé) pour prendre en compte le maximum de mots. C'est en fait une sorte de micro système expert adapté aux élèves des classes primaires.

Cet environnement est intéressant d'un triple point de vue :

- montrer un exemple de programmation déclarative (ou presque déclarative puisque l'ordre des règles intervient), c'est l'un des seuls environnements montrant l'intérêt d'une programmation déclarative, lié à des connaissances élémentaires (niveau primaire) et ne traitant pas d'une classification,
- comprendre le fonctionnement de certains aspects de la langue,
- montrer comment créer un logiciel de test capable d'expliquer ses réponses. (Montrer aussi concrètement qu'une machine peut écrire des âneries, en appliquant des connaissances fausses qu'on lui a rentré, et dont on peut retrouver l'origine). Le repérage de l'erreur peut se faire ici à l'aide d'un dictionnaire. L'annexe II décrit le fonctionnement interne du système.

Citons aussi MUSILOG [CAUBISENS 88] qui est un environnement sur la musique. FORMATEXTE [BRUILLARD 86c] un formateur de texte qui a diverses particularités comme d'intégrer complètement les fonctions Logo, disposer de fonctions de remplacement étendues (remplacer un caractère par un dessin, jouer sur les couleurs, etc.).

D'autres réalisations personnelles n'ont pas été éditées, mais sont néanmoins largement utilisées dans le Val de Marne et à l'Ecole Normale de Bonneuil.

LOGO-CONTE permet de construire un conte illustré à partir de LOGO. Le travail s'effectue en deux étapes :

- création en Logo standard de l'ensemble des personnages et des éléments de décor nécessaires à la réalisation de l'histoire,
- création des scènes incorporant les éléments construits dans la première phase. Cette deuxième phase ne nécessite aucune programmation, des utilitaires généraux permet-

tent d'exécuter un dessin à la taille désirée (sur le modèle de REPETE, deux procédures GROSSIS et DIMINUE sont disponibles), et on positionne la tortue directement avec le crayon optique. La scène créée est sauvegardée comme une image. On écrit, de plus, dans l'éditeur un texte ne dépassant pas une page. Les scènes et textes associés étant sauvés avec un numéro, le programme chaîne automatiquement ces éléments par ordre croissant. La production finale est une histoire que l'on peut découvrir sur l'écran ou imprimer.

LOGO-CONTE est un exemple d'environnement Logo destiné à favoriser la collaboration des enfants en vue d'une production. Chacun, en faisant un personnage, participe à la création collective, et de même est amené à incorporer le travail des autres dans la réalisation de sa scène.

MULTI-TORTUES : sorte de langage objet mono-classe (la classe 'tortues'), dont l'intérêt est de pouvoir manipuler directement plusieurs tortues (faire des courses, ou des poursuites, comme la courbe du chien, montrer les isométries, etc.).

CIBLES et FLECHES : environnement d'entraînements aux mesures de distance et d'angles. La spécificité est ici de pouvoir changer divers paramètres (en particulier taille de la cible et pas de la tortue), et de comparer divers trajets effectués.

MINILOGO : Logo où on pilote la tortue en appuyant directement sur des touches (A correspond à AV 10, D droite 30, etc.), mais avec la possibilité de créer de nouvelles touches (conserver l'esprit du Logo).

Au delà de cette énumération, il est intéressant de constater que les plus utilisés dans les classes sont :

- LOGO-CONTE, essentiellement parce qu'il permet des travaux en collaboration (voir le thème général de collaboration),
- MINILOGO, car il permet de travailler avec des élèves plus jeunes (CP ou CE1),
- CIBLES, car il se présente comme un jeu.

PROD souffre d'une inéquation à une structure type nano-réseau. Sa place est plus dans la classe, comme une mémoire collective, que l'on consulte, corrige et complète au fur et à mesure que l'on rencontre de nouveaux mots. Avec MULTI-TORTUES, il sont essentiellement utilisés en formation, et permettent d'explorer de manière simple d'autres paradigmes de programmation que la programmation procédurale.

II.1.c Tuteurs (dits intelligents)

Ce chapitre décrit l'autre versant de l'utilisation des ordinateurs en formation, i. e. l'intégration d'une interaction de type prioritairement tutorielle où la machine tente de jouer le rôle du maître. On replacera les tuteurs dans la perspective plus globale de l'interaction homme-machine et on essayera de faire ressortir les obstacles fondamentaux d'une telle entreprise.

II.1.c.1 Architecture générale des tuteurs

Des articles de synthèse sur le domaine de l'EIAO proposent immanquablement des architectures générales de tuteurs intelligents. On peut citer, entre autres : [SLEEMAN & BROWN

82] [ROSS 87] [YAZDANI 87] [WENGER 87] [BURNS & CAPPS 88] [NICAUD & VIVET 88]. L'idée générale tourne autour des trois composantes qui interagissent dans la formation : le sujet, l'élève et le professeur, i. e. quoi, qui et comment (" *what, who, how* " [SELF 74]).

Hartley et Sleeman [HARTLEY & SLEEMAN 73] ont les premiers spécifié les divers ingrédients d'un tuteur intelligent :

- connaissance du domaine,
- connaissance de la personne qui reçoit l'enseignement (modèle élève),
- connaissance des stratégies d'enseignement,
- connaissance de la façon d'appliquer la connaissance des stratégies d'enseignement aux besoins d'une personne.

La conception dominante organise les tuteurs intelligents autour de quatre modules :

- le domaine,
- le modèle élève,
- l'expert pédagogue,
- l'interface.

Divers autres modèles généraux ont été proposés, s'écartant de manière plus ou moins importante du modèle général. O'Shea [O'SHEA & Al. 84] décrit un modèle constitué de 5 anneaux :

- histoire de l'élève,
- modèle de l'élève,
- un ensemble de stratégies d'enseignement,
- un générateur d'exercices qui orchestre l'information des trois modules précédents pour produire les sorties,
- un administrateur responsable du contrôle général. Dans cette architecture, la connaissance du domaine n'est pas explicitement représentée.

Anderson, concevant des tuteurs sur la base de sa théorie ACT, exclue le modèle de l'élève en le remplaçant par un catalogue des erreurs possibles [ANDERSON & REISER 85].

Wenger [WENGER 87] reprend le modèle traditionnel dans une perspective de communication :

- le domaine est l'objet de la communication,
- le modèle élève caractérise le récipient de la communication,
- le modèle pédagogique gère l'aptitude à la communication,
- l'interface correspond à la forme de la communication.

Burns et Capps [BURNS & CAPPS 88] précisent certaines caractéristiques des différents modules :

- Module expert (sur le domaine),
- Module de diagnostic de l'élève,
- Module gérant le curriculum et l'instruction :

- contrôle de la représentation de la connaissance pédagogique pour choisir et séquencer la matière,
- aptitude à répondre aux questions de l'élève sur les buts de l'enseignement et son contenu,
- stratégies pour déterminer quand les élèves ont besoin d'aide et fournir l'aide appropriée.
- Environnement d'instruction,
- Interface homme-machine.

Les divers modules peuvent être de plus répartis en différents sous-modules dont le découpage n'est pas toujours cohérent avec l'organisation standard.

On distingue ainsi le 'tutoring' du monitoring' (DELTA / TMIE) : 'Tutoring' : Ensemble des décisions prises et des actions exercées par les intervenants formateurs au sein d'un environnement d'apprentissage dans le but d'agir sur les conditions de travail de l'apprenant pour optimiser son apprentissage. 'Monitoring' : Procédure d'observation, de recueil, de traitement et de mise en forme d'information en provenance de l'environnement d'apprentissage, sur les caractéristiques et les comportements des différents éléments constitutifs, utiles aux différents niveaux de régulation du système évaluatif.

Le modèle classique correspond souvent peu aux implantations réelles, et les concepteurs d'architecture générique sont amenés à en limiter l'intérêt. Ainsi, Self [SELF 88] souligne que la division d'un tuteur intelligent en modules domaine, élève et tuteur est plus une forme d'explication qu'un guide pour l'implantation.

D'un autre côté, on admet des définitions plus fonctionnelles, évitant les interprétations abusives d'implantation. Miller [MILLER 88] décrit ainsi un ITS comme la conjonction de 3 choses :

- un instrument éducatif,
- une interface à un programme d'application,
- un système basé sur la connaissance.

II.1.c.2 Interaction Homme-Machine / Systèmes d'aide

On peut situer le problème des tuteurs dans le cadre plus général de la communication homme-machine. Cette dernière recouvre en fait :

- les tuteurs,
- les systèmes d'explication,
- la documentation informatisée,
- l'aide en ligne.

Tous ces domaines ont en commun les problèmes généraux d'interface homme-machine et de modélisation de l'utilisateur. Les liens sont d'ailleurs très proches et les frontières difficiles à dessiner :

- explication dans les systèmes experts,
- contrôle dans les environnements d'apprentissage,

- actions tutorielles locales dans le cadre de la résolution de problèmes,
- tuteurs intégrant explications, aide et documentation.

Wenger [WENGER 87] cite les systèmes d'aide, les interfaces utilisateurs, la recherche dans les bases de données, les systèmes experts (explications, justifications).

En particulier, dans des domaines déjà informatisés, s'appuyant sur ou étant directement liés à l'informatique (enseignement de la programmation, utilisation d'outils généraux : traitement de textes, publication assistée par ordinateur, logiciels de gestion, etc.), une liaison peut s'établir avec le cadre de travail, évitant la séparation entre utilisation et formation. L'aspect apprentissage proprement dit se confond dans l'interaction même avec les autres aspects inclus dans l'environnement de travail (documentation, aide, explication). Ce phénomène s'étend en fait au fur et à mesure de l'intrusion de l'informatique dans les processus de travail dans tous les domaines où cet outil intervient. Dans un cadre de formation continue, on pense à des sessions discontinues de suivi d'un tuteur (III.2, page 91). Il y a souvent des difficultés dans l'usage des diverses fonctions d'un logiciel, les concepts associés étant mal maîtrisés, nécessitant des interventions tutorielles. Les différences entre ces dernières et les systèmes d'aide proviennent alors surtout du but poursuivi et de l'organisation globale.

Cette vision communication homme-machine inclut des formes de communication implicites et explicites [FISCHER & Al. 84]. Elle nécessite 4 domaines de connaissance :

- le domaine lui-même,
- le processus de communication,
- le partenaire de la communication,
- les problèmes classiques rencontrés par les utilisateurs.

Cette architecture n'est finalement pas très éloignée de celle proposée pour les tuteurs. Il faut remarquer néanmoins une référence explicite aux problèmes rencontrés par les utilisateurs, vision pragmatique (proche du catalogue d'erreurs de Anderson V.1.b, page 152) qui exclue la réalisation d'un diagnostic automatique non basé sur des protocoles d'observation (V.1.c, page 154). Cette idée est développée dans EUROHELP [BREUKER 88] dans la production d'une typologie des questions dérivées des expériences des utilisateurs travaillant sur des cas complétée par des questions posées aux experts (difficultés, besoins de clarification, etc.). Voir aussi [WAERN 84].

Le cas des systèmes d'aide est exemplaire, dans ses parallèles avec les tuteurs intelligents et dans la prise en compte fine des besoins des utilisateurs. Boulet et son équipe [BOULET & Al. 89] cherchent à développer un logiciel générique pour créer et implanter des systèmes 'conseillers' en général et à déterminer des méthodes pour l'application d'un tel système à des domaines spécifiques et aux requêtes des utilisateurs. Ils distinguent trois modes d'utilisation :

- mode réactif : offrir des réponses à l'utilisateur,
- mode proactif : observer ('monitor') l'utilisateur et intervenir en fournissant des suggestions pour l'aider à réaliser son but,
- mode tutoriel : faire découvrir le système à un néophyte.

On différencie d'ailleurs les systèmes d'aide dits passifs (interface par touches ou langage naturel), qui attendent les sollicitations de l'utilisateur et les systèmes d'aide actifs qui observent l'activité de l'utilisateur et sont susceptibles d'intervenir sans requête de ce dernier.

Boulet [BOULET & Al. 89], dans le cadre du module d'intervention du système conseiller, utilise un schème à facettes multiples comme modèle de représentation des connaissances et de l'usager. On retrouve les mêmes idées dans le cadre du projet EUROHELP [WINKELS & Al. 88] au niveau de la conception d'un 'coach' générique :

- aider l'utilisateur à résoudre un problème courant,
- enseigner à l'utilisateur la maîtrise des IPS ('Information Processing Systems', i. e. programmes informatiques interactifs).

De nombreux systèmes sont ainsi développés, dans un cadre très informatique, plus comme des outils d'aide intelligents que des tuteurs :

- langages de programmation (COMMON-LISP [SELKER 89]),
- systèmes d'exploitation (UNIX [DANLOS & Al. 85], projet EUROHELP).

Contrairement aux tutoriels, les systèmes d'aide ne peuvent pas être structurés à l'avance, mais doivent «comprendre» les contextes spécifiques dans lesquels l'utilisateur demande ou a besoin d'aide (de même pour les hypertextes, voir III.2.c et III.3.a.3, page 106). Dans les environnements permettant une apprentissage par la découverte guidée, il faut pouvoir combiner l'exploration et un système d'aide actif [HENNESSY 89]. Ceci impose l'utilisation de techniques de reconnaissance de plans, et plus généralement de techniques de diagnostic (passage du comment au pourquoi, voir V.1.d, page 159).

II.1.c.3 Interface - Module Environnement

Dans les diverses architectures proposées pour les tuteurs intelligents (II.1.c.1, page 37), on remarque l'apparition tardive de l'interface. Elle a souvent été négligée, ce qui peut s'expliquer par différentes raisons :

- les interfaces graphiques sont assez récentes, les premiers développements de l'EIAO n'ont pu les prendre en compte.
- la conception d'une interface est un travail long et difficile : Donadi [DONADI & Al. 89] rapporte que lorsque la refonte de l'interface de Wordstar a été entreprise pour produire Wordstar 2000, l'effort a été majeur, quasiment équivalent à l'écriture d'un programme entièrement nouveau. Ainsi, le travail requis pour passer d'une maquette à un produit ne se justifie que pour une expérimentation ou une diffusion.
- Une conception erronée considérant l'interface comme transparente, offrant une plus grande simplicité d'accès mais ne changeant en rien les caractéristiques essentielles d'un programme.
- l'orientation même des recherches qui prend l'enseignant humain comme modèle ce qui bute sur l'ensemble des phénomènes que la machine ne peut prendre en compte et néglige de considérer les spécificités de ce nouveau medium.

La tendance est en train de s'inverser, le travail sur l'interface et sur l'environnement de travail devient prépondérant. En fait, les travaux débouchant sur des réalisations concrètes prennent nécessairement en compte l'utilisateur de manière importante, non comme un être abstrait idéal, mais comme un des éléments centraux du système. On essaye ainsi d'adapter la technologie aux contraintes de cet utilisateur et non à partir de l'imitation d'autres modes de communication.

L'interface est centrale dans le sens où elle est l'unique mode d'accès à l'environnement, et le module environnement « *définit les types de problèmes que l'élève a à résoudre et les outils dont il dispose pour le faire* » [BURTON 88].

Tout d'abord, l'apport des écrans graphiques, souris, icônes, menus déroulants, etc., a considérablement amélioré les interfaces, dont la prise en main est grandement facilitée (idée de 'look and feel'). Les interfaces de manipulation directe se caractérisent par [SHNEIDERMAN 82], [SHNEIDERMAN 83] :

- une représentation continue des objets d'intérêt,
- des actions physiques ou des pressions sur des boutons étiquetés à la place d'une syntaxe complexe,
- des opérations rapides, incrémentales et réversibles dont l'impact sur les objets est immédiatement visible.

On peut séparer trois différents aspects de la manipulation directe [HUTCHINS & NORMAN 85] :

- mimétique : la forme physique des actions qui sont exécutées imite les opérations désirées dans le domaine considéré,
- sémantique : il existe un couplage étroit entre l'interface et le domaine (« *construire la sémantique du domaine dans le comportement de l'interface* »), i. e. les modèles conceptuels du domaine et de l'interface doivent être aussi similaires que possibles,
- temporelle : rétroaction immédiate à propos des actions, du modèle conceptuel et de ce qui est perçu comme résultat. Les systèmes peuvent être ainsi considérés comme étant directement manipulables dans l'un, deux ou tous les sens précédents.

On distingue aussi les interfaces à la première personne et les interfaces à la seconde personne :

- menus,
- langages de commande,
- langage naturel.

L'interface de CABRI-GEOMETRE par exemple est essentiellement à la première personne, complétée par un système de menus déroulants. Elle est basée, comme de nombreux systèmes graphiques de ce type, sur la notion de contrainte.

Ensuite, on reconnaît le fait que savoir accéder aux connaissances est une connaissance qui est primordiale (voir III.1, page 77), et que simplifier et visualiser ces accès amènent une réification de la connaissance qui est un des atouts majeurs des ordinateurs. Dans cet esprit, l'une des choses les plus ardues « *est de trouver une architecture qui puisse servir de représentation mentale pour les intentions de l'utilisateur* » [ROHR & TAUBER 85].

"*The advantage of natural language for beginners, of course, is that most of them already know how to use it moderately well. Natural language, however, is verbose, vague and ambiguous, and if a system uses any natural language at all then a user may easily overestimate the system's language-understanding capabilities. Helping users to appreciate the extent to which computer dialogues are restricted is one of the hardest problems in interactive system design.*"

[SELF 85a]

L'utilisation d'autres techniques nécessite de surmonter le même écueil, i. e. de faire comprendre aux utilisateurs ce dont est capable le système et ce qu'ils peuvent faire avec. Un échec amène soit une sous-utilisation soit, au pire, un rejet complet du système. Cet aspect est d'autant plus complexe, que de nombreux utilisateurs refusent de faire l'effort de comprendre le système et confondent son fonctionnement avec l'image qu'ils en ont qui est le plus souvent liée à des préconceptions (voir les problèmes d'usage).

Lewis [LEWIS & Al. 87] décrit ainsi les contraintes que doit respecter une interface d'un programme d'EIAO ('Anderson Teacher's Apprentice') :

- Aussi facile que possible à utiliser, i. e. minimiser le nombre d'actions nécessaires pour communiquer avec le tuteur.
- Une structure ou représentation aussi congruente que possible à la structure sous-jacente des problèmes à résoudre.
- Etre hautement interactive et fournir le plus d'information possible sur les étapes intermédiaires de résolution.
- Pouvoir pointer les erreurs de bas niveau quand elles arrivent ('monitor' continu)
- Pouvoir faire varier le chargement de la mémoire de travail : donner accès aux informations reliées au problème (voir III.1, page 77).

Les techniques de visualisation (avec ou sans possibilités d'animation) sont de même importantes car elles facilitent l'émergence de modèles mentaux. En partie métaphores, analogies, symboles, l'important n'est pas de savoir si elles sont vraies, mais si elles sont utiles (perspective pragmatique). On peut citer ainsi l'utilisation de points de vue [MOYSE 89] (voir III.2.b.5, page 98), de diagrammes [HALL 89], d'animation [NATHAN & Al. 89], ou de formes de tableau de bord (GUIDON-WATCH [RICHER & CLANCEY 87]).

Elles conduisent à un réexamen des modes de pensée, voire même à une mise en cause des modes de codage ou représentation traditionnels imposés par l'histoire et dont la justification ne tient plus qu'à la tradition. Une difficulté reconnue dans de nombreux domaines nécessitant de résoudre des problèmes est que la syntaxe d'une solution ne reflète pas le processus de raisonnement requis pour la construire [REISER & Al. 89] (ce qui conduit à l'idée d'une refonte des notations habituelles [KAPUT 86]).

Le travail sur DYNABOARD [KALTENBACH & FRASSON 89] va ainsi dans ce sens : « *Comme des analogies concrètes appropriées sont difficiles à trouver pour les formes abstraites de raisonnement mathématique, il faut traiter directement ce symbolisme abstrait... Des facilités informatiques doivent être développées pour permettre de transformer graduellement des énoncés mathématiques en une terminologie plus familière et réciproquement.* » (Voir le paradigme des deux mondes, II.2.a.3, page 54)

On retrouve les mêmes problématiques que celles des systèmes hypertextes dans le recours au collage (i. e. lien de remplacement en GUIDE, III.1.a.2, page 79), et l'utilisation d'icônes (voir III.1.a.3.b, page 82). « *Il y a une progression continue de systèmes qui aident à clarifier la connaissance générale, vers des systèmes qui participent activement au développement de nouvelles connaissances humaines.* »

Ce type de techniques semble être une alternative aux techniques difficiles issues de l'IA : [EISENSTADT 89]
" *..., the role of visualisation techniques for teaching programming may prove to be more effective than theories of student-modelling, cognitive 'plans' and automatic program debugging.*

Also, the virtues of a well-structured curriculum (necessary for providing good debugging advice) must be weighed against the benefits of an opened-ended environment (necessary for foster creative learning and experimentation). "

Dans les comparaisons effectuées entre les machines et les humains, il faut prendre en compte les spécificités des modes de communication :

- oralité, le paradigme du cours traditionnel est complétée par des modes de communication non verbaux (intonation, gestes, etc.),
- texte imprimé (taille, disposition, références croisées),
- texte/graphique/animation.

Ces nouveaux modes de communication, même s'ils sont prometteurs, sont encore inhabituels et nécessitent de développer des stratégies et des aptitudes particulières. Nous en sommes encore dans une phase de conception et d'appropriation ce qui nous confronte à des obstacles culturels qui freinent leur utilisation (voir III.3, page 103). Dans les premières expériences sur les hypertextes, on constate que les étudiants ne réussissent pas mieux qu'avec les livres et qu'ils trouvent ces derniers plus faciles à utiliser. Mais cette constatation doit intégrer le fait que le livre est un objet culturel depuis plusieurs siècles alors que l'hypertexte n'existe que depuis quelques années! (Voir III.1.b.2, page 87)

II.1.c.4 Réalisation des tuteurs intelligents : problèmes généraux

A l'heure actuelle, aucun tuteur construit sur le modèle défini plus haut (II.1.c.1, page 37) n'est réellement opérationnel. Les nombreux travaux déjà parus focalisent soit sur un des aspects (domaine, élève, tuteur), soit sur la globalité en restreignant les contraintes, mais seules les architectures 'papier' répondent aux exigences formulées. Cet état de fait suscite des questions sur les directions de recherche à suivre qu'on ne peut esquiver. L'architecture proposée est-elle réaliste? Est-ce un but quasiment inaccessible dans les quelques années à venir, et n'est-ce pas alors un modèle trop idéal bloquant d'autres pistes fécondes? Si l'objectif visé est réellement l'apprenant, des architectures moins ambitieuses ne permettent-elles pas de répondre de manière satisfaisante?

Les obstacles à la réalisation des tuteurs intelligents peuvent être classés en trois grandes catégories :

- **difficulté de réalisation d'un des modules,**
- **articulation entre les modules,**
- **intégration du tuteur dans un cursus réel.**

Nous allons les passer en revue.

II.1.c.4.a Problèmes liés aux différents modules

Au niveau de l'expertise du domaine, on est face à de redoutables problèmes de représentation de connaissances. Il n'y a souvent pas de formalisation (ou elle est incomplète ou partiellement fausse, ou trop complexe pour être enseignée). L'enseignement génère d'ailleurs ses propres formalisations floues génératrices d'erreurs (voir par exemple IV.1.a, page 113). L'existence même de résolveurs, au lieu d'apporter une solution, déplace les problèmes à résoudre (le

résolveur lui-même devient objet d'enseignement, II.2.c, page 65). On constate, à partir des tentatives de représentation, qu'on dispose de peu de connaissances, que ce soit au niveau du domaine proprement dit, qu'au niveau méta. La règle est en fait de travailler avec des résolveurs incomplets (IV.2.d, page 132).

La modélisation de l'élève bute sur les mêmes difficultés. On sait encore peu de choses sur les erreurs faites par les élèves, encore moins sur leur origine et les façons d'y remédier (voir par exemple, la conclusion d'Odile Paliès [PALIES 88a] dans son travail sur le diagnostic cognitif et V.3.b, page 173).

La conception même d'un tuteur artificiel suppose de pouvoir répondre à des questions encore largement ouvertes. Comme le souligne A. Bork [BORK 88], nous n'avons pas de théorie effective sur la façon dont les élèves apprennent, et il est peu vraisemblable que de telles théories puissent émerger rapidement. On ne dispose que de bribes de théories différentes et difficilement conciliables. [WINANS & Al. 88] montrent ainsi six théories et leurs implications dans le cadre des actions d'un tuteur (quand intervenir? Que dire? Que faire ensuite? (Skinner, Gagné, Bandura, Weiner, Piaget et 'information processing theory'). Certaines méthodes suggérées semblent ardues pour un tuteur artificiel, d'autres, a contrario, sont difficiles voire impraticables pour un tuteur humain mais peuvent être réalisées facilement avec une machine (exemple du renforcement recommandé par Skinner).

En ce qui concerne l'expertise pédagogique, les trous sont flagrants [DILLENBOURG & GOODYEAR 89] :

- Pas de théorie cohérente et complète du 'tutoring' (seulement des fragments),
- La connaissance tutorielle est partiellement spécifique du domaine,
- Beaucoup de stratégies utilisées semblent efficientes mais les théories associées sont incapables de trouver une explication rigoureuse pour être formalisée de manière calculatoire.

L'implantation de plans pédagogiques est déjà complexe [VIVET & Al. 88] .

Il y a encore peu de recherches sur le tutorat réel : on commence à prendre en compte certains éléments cognitifs, mais pas les éléments affectifs. Maintenir la motivation est un objectif non atteint [LEPPER & CHABAY 88]. L'expertise tutorielle n'existe pas, les enseignants travaillent avec des groupes et font très peu de tutorat (où sont les experts?).

Atkinson [ATKINSON 76] a montré que l'apprenant n'est pas particulièrement un décisionnaire efficace pour son propre processus d'apprentissage, mais cette gestion du curriculum reste encore très peu maîtrisée.

Les interfaces évoluent très rapidement (par nécessité économique, pour permettre au plus grand nombre de personnes d'accéder aux ordinateurs ce qui contraint à rendre leur maniement le plus immédiat possible). Par contre coup cela ouvre de nouvelles perspectives à l'utilisation éducative des machines, mais induit des incidences déterminantes sur la production des programmes d'EIAO. Ces derniers nécessitent le recours à des équipes de plus en plus larges et à des moyens importants. On passe des langages auteurs aux ateliers de production intégrant des équipes de développement. En particulier, les systèmes sont forcément moins génériques (IV.3.c.1, page 140). Il y a plus d'intérêt sur les aspects multi-média que sur des techniques de modélisation (la gestion du multimedia, qui ne peut se concevoir comme une simple addition de média, n'a rien d'immédiat).

II.1.c.4.b Problèmes d'articulation des différents modules

Même si localement, on arrive tant bien que mal à des choses significatives sur chacun des modules, leur coopération est loin d'être assurée : on a l'impression d'être en face d'un jeu de MECCANO dont les pièces ne s'emboîtent pas correctement. De bonnes recherches en IA transforment les optimistes en dubitatifs, dès qu'on les sort de leur frange étroite d'applicabilité, elles se dégradent rapidement en dehors de leur sphère de compétence, incapables de combiner différentes sortes de connaissances [du BOULAY & SLOMAN 88]. C'est un problème classique en IA, où des réussites locales ne sont ni garantes ni forcément des amorces de solutions plus globales. Préciser les contraintes sur chacun des modules pour qu'ils puissent communiquer avec les autres : Qu'est-ce qui est dépendant et qu'est-ce qui est indépendant? Les liens entre les modules sont multiples :

- modèle élève - théorie de l'apprentissage : l'apprentissage cumulatif (Gagné) correspond mieux à un modèle d'expertise partielle ('overlay', V.1.b, page 152), tandis qu'un apprentissage constructif réclame plutôt un modèle intégrant les erreurs ('buggy model' V.1.b, page 152) [OHLSSON 86].
- le module expert détermine le module environnement et induit certaines techniques de diagnostic (V.1.c.6, page 158), etc.

Cette fausse indépendance (modularité en fait abusive), a des répercussions sur la pertinence des environnements proposés : des formes d'apprentissage ne seront elles pas favorisées de par leur plus grande facilité de mise en oeuvre technique (réductions mal maîtrisées dans des projets irréalistes)? Les tâches dans le monde réel ne ressemblent pas aux problèmes scolaires. Typiquement, chaque ensemble de problèmes, à l'école, est soigneusement structuré pour enseigner une habileté particulière, plus ou moins isolée des autres habiletés [COLLINS & BROWN 86]. Par contre, pour gérer un modèle d'élève permettant de gérer un cursus, un découpage de ce type semble plus facile. Par opposition, une pédagogie basée sur la réalisation de projets dans lesquels il n'y a pas de but bien défini ou de méthode pour aboutir à la solution risque d'être exclue.

II.1.c.4.c Problèmes d'intégration

Un des problèmes majeurs qui ne peut être évité est celui du mode d'utilisation des tuteurs artificiels dans les organisations de formation existantes. Les processus cognitifs sont entièrement interconnectés à l'organisation sociale de l'instruction [NEWMAN 89]. Ainsi, le public visé (formation initiale obligatoire, formation continuée) et les structures mises en place conditionnent de manière déterminante la façon dont les logiciels destinés à la formation sont et seront utilisés (II.3.a.5, page 71).

On est face à plusieurs questions :

- compétence des tuteurs (sur le plan tutoriel) : leur limitation est très importante à cerner pour éviter les blocages et assurer une bonne complémentarité avec d'autres modes d'enseignement.
- problèmes organisationnels : comment transformer les structures existantes pour inclure ces nouveaux outils.

- problèmes de RESPONSABILITE : il n'est pas concevable à l'heure actuelle, de confier la gestion totale d'un apprentissage quelconque à des machines. Des personnes (physiques ou morales) en ont toujours la charge réelle.
- problèmes d'usage qui sont liés à l'image qu'en ont les enseignants, les prescripteurs et les élèves.

Remarque : dans les tests effectués avec les logiciels créés avec SEVE (III.2.b, page 92), celui qui a été le mieux accueilli par les utilisateurs est celui destiné à la formation continue, dans un cadre organisationnel spécifiquement défini (III.2.b.4, page 97).

Si on regarde du côté des besoins, on s'aperçoit que le champ des tuteurs intelligents est en fait pris par d'autres applications. En effet, le tutorat suppose une organisation spécifique et impose une certaine durée. L'utilisation d'outils en-ligne rend l'adéquation aux besoins plus problématique : est-il nécessaire d'être spécialiste ou expert dans de nombreuses disciplines :

- Systèmes Interactifs d'aide à la décision [LEVINE & POMEROL 89] (modèle élève, choix, interprétation des outils). La question est plus de savoir comment utiliser ces divers outils de manière optimale (s'approprier une part d'expertise sans se transformer en expert).
- curriculum : accès à des banques d'information (III.1.a, page 77),
- systèmes d'aide (II.1.c.2, III.3.a.3, page 106).

Les cas où l'apprentissage semble indispensable s'inscrivent dans des organisations préétablies, ayant une certaine méfiance vis-à-vis de solutions trop techniciennes. Le problème est bien d'intégrer ces divers outils dans la démarche globale, non pas de faire assurer à une machine l'ensemble du processus (outils d'évaluation, de diagnostic, de présentation des EAO ou EIAO locaux, etc.). L'évolution conduit d'ailleurs plus vers des systèmes analogues aux SIAD (systèmes interactifs d'aide à la décision), i. e. des systèmes interactifs d'aide à l'enseignement (voire des SIAA, systèmes interactifs d'aide à l'apprentissage).

La notion de tuteur intelligent semble évoluer vers de nouveaux paradigmes :

- l'idée de collaborateur potentiel [GILMORE & SELF 88], [CABROL & Al. 87],
- des environnements d'apprentissage qui supportent certaines formes d'apprentissage par la découverte guidée [HENNESSY & Al. 89],
- l'"apprenticeship" [NEWMAN 89].

Ce dernier modèle défend une forme d'apprentissage par l'exemple :

" Our design for intelligent instructional systems are based on the notion that students can come to understand the expert approach to the problem by observing examples of expert problem-solving and by seeing how their actions are interpreted within the framework of the expert understanding. " (idée proche de CAMELEON, VI.3.c, page 196)

Les instructeurs humains font partie du processus d'entraînement en interprétant les feed-back et suggérant de nouvelles pratiques. Il s'agit de mettre le pouvoir dans les mains de réels instructeurs et des élèves.

" The use of ITSs in the context of human instructor-student interaction provides extensive human ressources (instructor and students) for monitoring progress and directing next steps making some ITS features unnecessary. "

Cette vision plus pragmatique met en lumière les difficultés principales de l'introduction de ce type d'outil :

- compréhension de l'environnement, de ce que fait et que l'on peut faire avec la machine,
- compétence des instructeurs et la définition de leur rôle.

Chapitre II.2

Mathématiques et EIAO

Nous allons passer en revue un certain nombre de systèmes dans le domaine des mathématiques, en prenant successivement trois points de vue :

- une vision historique retraçant l'évolution des idées,
- un classement suivant les contenus mathématiques abordés,
- une comparaison des diverses architectures utilisées.

Nous terminerons en dégageant les problématiques générales de développement de systèmes intelligents dédiés à l'enseignement des mathématiques.

II.2.a HISTORIQUE

L'une des caractéristiques essentielles des logiciels d'EIAO étant la capacité de pouvoir résoudre les problèmes posés à l'apprenant, l'évolution de ces systèmes est directement liée aux contenus mathématiques maîtrisés par les machines. Les logiciels d'EAO ont d'ailleurs rapidement tiré parti des possibilités de calcul numérique des ordinateurs en les intégrant dans les tests. Ainsi, une première ligne de force est liée au développement de la démonstration automatique de théorèmes (DAT) et des logiciels de calcul algébrique. Une tradition plus EAO et associée aux recherches des psychologues et didacticiens sur la modélisation des apprenants s'est intéressée aux erreurs et aux conceptions erronées ('misconceptions'). Enfin, l'approche Logo ou «micromondes» est d'une certaine façon mitoyenne, centrée sur le processus d'apprentissage et l'activité de l'apprenant.

II.2.a.1 D. A. T., Calcul Algébrique et Systèmes Experts

Pour aller au-delà du calcul numérique, on peut distinguer deux voies ([VIVET 84]) : la voie de la logique, de l'inférence, de la déduction, de la démonstration automatique de théorèmes, et celle du calcul algébrique. On peut leur associer deux approches différentes : l'utilisation d'algorithmes et la programmation heuristique.

Un bref historique de la D. A. T. peut être trouvé dans [PASTRE 84] :

- le 'sequent calculus' [WANG 60],
- le principe de résolution de Robinson [ROBINSON 65],
- l'algorithme d'unification [PITRAT 66],
- la déduction naturelle [BLEDSOE 71] (dans le système PROVER),
- l'utilisation de méthodes graphiques.

En ce qui concerne les systèmes de calcul formel (voir [VIVET 84], [DAVENPORT & Al. 87]), on peut citer quelques réalisations importantes : MACSYMA [MARTIN & FATEMAN 71], REDUCE [HEARN 71], MAPLE, SCRATCHPAD, muMATH, MATHEMATICA [WOLFRAM 88], DERIVE. La plupart ne sont accessibles que sur des gros systèmes ou des stations de travail, on peut cependant noter que depuis muMATH (utilisable sur IBM-PC) des versions performantes apparaissent sur micro-ordinateur (MATHEMATICA sur MacIntosh, DERIVE sur IBM-PC).

Les deux approches (algorithmique, heuristique) ont coexisté et coexistent encore, parfois sur le même domaine (exemple de l'intégration [VIVET 84]). L'approche algorithmique est essentiellement théorique, tandis que l'approche heuristique est plus expérimentale. Cette dernière s'inscrit dans la voie de recherche des systèmes experts [LAURIERE 87]. La vision de CAMELIA [VIVET 84] est un essai de combinaison des deux approches. L'apport de l'algorithmique à l'EIAO se situe principalement dans la mise à disposition de nouveaux outils éventuellement utilisables à des fins d'enseignement. L'approche de la programmation heuristique s'avère beaucoup plus fructueuse. Elle se base sur un 'dialogue homme-machine' qui doit permettre :

- *« d'une part de recueillir aisément la connaissance des spécialistes; ceci pose en particulier un problème de langage.*
- *d'autre part une explication par le système du raisonnement suivi, dans un but pratique de contrôle, d'aide ou de formation; ceci pose le problème du mode de raisonnement. »* [BARON 82]

Ainsi, les connaissances intégrées dans les systèmes s'appuient sur l'observation de mathématiciens dans des tâches de résolution ([BUNDY 75], [PASTRE 78], [PASTRE 85]), qui est souvent une auto-observation. Cette démarche amène d'ailleurs souvent à la formulation de connaissances soit inconnues soit éparpillées. Elle permet de mieux comprendre le rôle de la connaissance implicite ('tacit knowledge') dans la résolution de problèmes.

D'après Dominique Pastre [PASTRE 84], on peut tirer diverses constatations de l'observation des mathématiciens :

- importance des connaissances accumulées par le mathématicien, et de tout un savoir-faire mathématique,
- importance des représentations,
- nécessité de travailler avec un langage non formel dans lequel on puisse exprimer des idées, des concepts, des méthodes. Ces points fournissent un éclairage très important sur la façon de procéder des experts qui doivent guider la réalisation de logiciels d'EIAO (disposer de nombreuses connaissances déclaratives et procédurales, intérêt des représentations intermédiaires et de langages non formels).

Le deuxième but assigné au dialogue homme-machine dans la démarche système expert est l'une des caractéristiques essentielles qui oriente cette démarche vers l'enseignement : la

possibilité pour la machine de fournir une trace de son raisonnement, i. e. la métaphore de la boîte de verre par opposition à la boîte noire des programmes algorithmiques ('black-box and glass-box experts' [GOLDSTEIN & PAPERT 77]).

On peut distinguer plusieurs générations dans les systèmes experts. La première génération prouve la faisabilité de cette approche dans différents domaines. Sur le plan mathématique, on peut citer entre autres l'étude des limites (DALI [LAURENT 72]), les groupes ([GILLET 79]), la sommation de combinaisons (SEME [BARON 82]), etc. Dans ces systèmes, les explications qui peuvent être fournies s'adressent avant tout à l'expert et ne sont pas adaptées à un utilisateur néophyte dans le domaine ou n'ayant pas encore les connaissances suffisantes. La trace accessible ne rend compte que du cheminement du système et n'explique pas le pourquoi des choix effectués. On est ainsi conduit à un modèle d'enseignement magistral où l'obtention d'un résultat justifie a posteriori les choix effectués pour y parvenir. L'histoire de GUIDON conçu à partir de MYCIN illustre bien les insuffisances de cette approche (voir VI.2).

La deuxième génération de systèmes experts (voir [KASSEL 89]) vise à pallier à ces problèmes en améliorant leurs capacités d'explication. Ceci implique de rendre explicite la stratégie de contrôle et les règles heuristiques, i. e. de pouvoir traiter la métaconnaissance. On en arrive à des systèmes qui peuvent contrôler leur résolution et leur trace. En particulier, un résolveur contraint doit pouvoir n'utiliser que des connaissances censées être connues par l'apprenant, et être capable d'adapter le détail de la trace au niveau supposé de l'apprenant (tuteur algèbre [FERRET & JIMENEZ 87], APLUSIX [NICAUD 88], NAIADE [JOAB 88], etc.). La trace elle-même est un objet mathématique intéressant qui peut être présenté à l'apprenant sous diverses formes, historique de la résolution (pistes abandonnées comprises), reflet de la pensée de l'apprenant (ALGEBRALAND [COLLINS & BROWN 88]). Les objectifs didactiques se portent souvent vers la maîtrise des heuristiques en essayant d'éviter à l'apprenant de se perdre dans l'effectuation des calculs qui sont conduits automatiquement par la machine. On retrouve là une conception proche des micromondes. Le chapitre VI détaille cette approche et ses limites.

II.2.a.2 La science cognitive

Une autre caractéristique centrale dans l'EIAO est théoriquement de s'adapter à l'apprenant en s'appuyant sur un modèle construit dynamiquement. Dans l'EAO classique, basé sur le cycle présentation-question-analyse de la réponse- branchement [CROWDER 59], il n'y a pas à proprement parler de modélisation de l'apprenant, mais des tactiques de parcours prédéterminées associées aux réponses correctes ou erronées de l'élève. L'un des apports majeurs de la psychologie cognitive ('cognitive science') a été d'étudier en détail les processus cognitifs (GPS, Logic Theorist [NEWELL & SIMON 72], voir par exemple [MATHIEU & THOMAS 85]) à partir de l'observation de sujets engagés dans des tâches de résolution de problèmes. En particulier, ceci a permis une compréhension plus profonde de la notion d'erreur ('bug' ou 'misconception').

On distingue ainsi :

- Détection des lacunes dans les connaissances (WEST), associée à un modèle élève dit de superposition ("overlay"),
- Détection des idées fausses : MACSYMA ADVISOR

- Détection des erreurs systématiques : les travaux autour de BUGGY ont montré que de nombreuses erreurs ne sont pas dues à un comportement erratique des élèves, mais à l'application correcte de procédures fausses.

Des protocoles d'observation ont conduit aussi à mieux comprendre les erreurs faites par les élèves (opérations, fractions, décimaux, algèbre). Les travaux autour de la soustraction ont amené diverses réalisations :

- BUGGY : entraînement pour les enseignants,
- DEBUGGY : diagnostic en différé à partir de problèmes résolus par un élève,
- IDEBUGGY : version de diagnostic interactif,
- REPAIR THEORY : théorie sur l'origine de ces erreurs systématiques, conçues comme des réparations locales face à des impasses.

On trouvera un historique dans [WENGER 87] traduit dans [CUPPENS 88].

Les travaux de [YOUNG & O'SHEA 81] rejoignent les précédents mais se focalisent sur les erreurs les plus fondamentales. Des études identiques se sont poursuivies dans d'autres domaines :

- algèbre élémentaire [MATZ 82], LMS [SLEEMAN 82] [SLEEMAN 83], PIXIE [SLEEMAN 87b],
- les décimaux [CAWSEY 86] [RESNICK 84a].

Les bugs résistent à la détection et à la remédiation pendant des années. Pire, ils peuvent refaire surface spontanément après une remédiation apparemment réussie (cité dans [VANLEHN 90]). On réfléchit ainsi sur le niveau d'intervention associé à une erreur [CAUZINILLE-MARMECHE & MATHIEU 88].

De nombreux travaux s'attaquent aux différents problèmes de compréhension :

- problèmes additifs, soustractifs, multiplicatifs (VERGNAUD, etc., voir [MAYER 83])
- les 'word problems' (GREENO, DAVIS, etc.)
- le passage de l'arithmétique à l'algèbre (RESNICK, MATHIEU).

En fait, la plupart des études portent sur ce que D. Lacombe appelle les deux régressions [LACOMBE 88b] :

- la régression algorithmique (addition, soustraction),
- la régression algébrique : *« ils (les élèves) opèrent sur des écritures, ce qui les a privés du sens initial, et ils opèrent non pas d'après des véritables règles formelles, mais d'après des pseudo-règles de type juridique. »*

Ceci soulève le point fondamental qui est l'articulation entre le sens et les techniques. La maîtrise de ces dernières suppose l'acquisition de mécanismes automatiques, qui ne peuvent être contrôlés que par un retour au sens : il faut d'un côté s'éloigner du sens pour acquérir une certaine habileté mais de l'autre pouvoir y revenir à tout moment.

Exemple : une étude sur les 'WORD Problems'

Un problème classique qui a été très étudié est celui des élèves et des professeurs. Le sujet doit écrire une équation, utilisant les variables E et P, pour représenter l'énoncé : 'Dans un certain collège, il y a six fois plus d'élèves que de professeurs.' On demande d'utiliser E pour le nombre d'élèves et P pour le nombre de professeurs.

Une large proportion d'étudiants écrivent l'équation : $6E = P$ (25% de 160 'first-year engineering students'). (Voir [KAPUT & CLEMENT 79], [ROSNICK & CLEMENT 80], relaté dans [DAVIS 84]).

La première hypothèse pour comprendre la source de cette erreur consiste à penser que les étudiants recopient l'ordre des mots dans l'énoncé du problème. Cette explication est fausse, comme l'ont montré les études de Clement et de ses collaborateurs en changeant l'ordre des mots (le phénomène n'est pas affecté). Ils ont de même changé le contenu sémantique pour éviter le recours à une aide liée au contexte (on sait qu'il y a plus d'élèves que de professeurs dans les collèges).

Par exemple, voici un autre problème :
Ecrivez une équation utilisant les variables C et G pour représenter l'énoncé : 'Dans un restaurant, pour 4 personnes qui commandent du camembert, 5 personnes commandent du gruyère.' Soit C le nombre de camemberts commandés, et G le nombre de gruyère commandés. (résultat : 50% des étudiants écrivent 4C = 5G).

La deuxième hypothèse pour comprendre cette erreur est que les étudiants utilisent un mauvais cadre de référence ('frame'). Au lieu de considérer le cadre des équations à variables numériques, ils travaillent dans le cadre des étiquettes ou unités ('label frame'), où l'on écrit par exemple 1m = 100 cm. Dans ce cas, m et cm sont des étiquettes référant à un nom d'unité, non des variables numériques. Cette explication est consistante avec les observations faites. En outre, on constate que les étudiants qui font cette erreur d'inversion décrivent la situation de manière différente de ceux qui écrivent l'équation correcte. De plus, l'enseignement du comportement correct, qui conduit temporairement à la performance correcte, ne peut éliminer le cadre des étiquettes ('label frame'), du fait peut-être qu'il a son utilité, mais ne peut pas non plus empêcher le fait de le retrouver dans des cas incorrects.

En conclusion, des structures de représentation de la connaissance persistent de façon plus ou moins permanente en mémoire, les formes les plus anciennes entrant en conflit avec des versions ultérieures révisées ou corrigées avec lesquelles elles coexistent. Ceci montre la difficulté du problème de la remédiation.

On s'aperçoit d'ailleurs (voir [MAYER 83]), que dans les 'algebra word problems' standards, les sujets se rappellent beaucoup mieux des informations pertinentes que des détails superflus, ce qui peut s'interpréter par le fait que des schèmes guident le codage et la recherche des informations dans ce type de problèmes.

Remarque : les deux problèmes précédents ont été donnés à des normaliens première année. Le premier est très largement réussi (21 sur 25), 2 personnes donnent une simple inégalité ($E > P$), et 2 seulement inversent l'égalité ($6E = P$). Dans le deuxième problème, les réponses sont plus diversifiées, mais seules 7 personnes trouvent la bonne solution (3 personnes ajoutent la contrainte ($C + G = 9$). 9 personnes écrivent la relation inversée ($4C = 5G$), et 9 personnes ne donnent pas de relation ou des relations farfelues ($G = C + 1$, $C < G$, $C + G = 4 + 5$, $4C < 5G$).
Dans le premier cas, il semble qu'un contrôle sémantique s'effectue, de plus les choix sont peu nombreux. Dans le deuxième cas, le nombre faible de bonnes réponses indiquent une manque de maîtrise de la proportionnalité. Les discussions avec les normaliens confirment néanmoins les explications précédentes.

Enfin divers protocoles sur la résolution de problèmes soulignent les différences entre les novices et les experts, entre les élèves brillants et ceux jugés plus faibles. Cette idée était le

point de départ de LOGO. Citons brièvement, sur les problèmes de représentation les travaux de [SCHOENFELD 87], et ceux de [SCANLON & O'SHEA 88] en physique. Dans les comparaisons entre les élèves [CHI & Al. 89], l'auto-explication semble un facteur discriminant entre les bons et les mauvais (truc pédagogique classique consistant à faire expliquer à un élève comment il fait). *« Une interprétation est que les auto-explications consistent en la création de règles d'inférence qui sont des instanciations des principes et définitions introduits dans le texte. »*

II.2.a.3 L'approche "micromonde"

L'approche micromonde (II.1.b, page 30) consiste souvent en une tentative de sauvegarde du sens d'une activité (un micromonde mathématique fournit une sémantique dynamique à un système formel [THOMPSON 87]), travailler de manière conjointe dans un univers concret et un univers abstrait. Cette approche nécessite de pouvoir créer des objets de transition et les manipuler (choix : manipulation directe, langages?).

L'apport de LOGO aux mathématiques a été imaginé par Feurzeig [FEURZEIG 69] : apprentissage de et par la programmation, activités de modélisation. Un premier panorama sur l'utilisation de langages et d'outils a été dressé [ROSS & HOWE 81] (panorama sur 10 ans) (voir aussi LOGO ([HOWE & Al. 84]). Les avantages de LOGO comme environnement proviennent de ce qu'il permet :

- des projets d'élèves importants,
- la discussion et l'expérimentation réflexive,
- des interprétations et illuminations sur le sens [HOYLES 85]. Et, au niveau des contenus [FEURZEIG & Al. 81] :
- rigueur de pensée,
- compréhension de concepts généraux (variables, procédures, fonctions, récursivité) et capacité à les utiliser,
- maîtrise de la planification et de la décomposition en sous-problèmes,
- recherche des erreurs (voir II.3.a.3, page 69).

D'autres langages sont bien adaptés à la création de micromondes :

- SMALLTALK : NOLOG [JONES & THORNE 88],
- PROLOG [ENNALS 84].

En dehors du Logo traditionnel, la plupart des micromondes développés s'appuient sur ce que l'on peut nommer le paradigme des deux mondes, qui consiste à travailler en parallèle ou de manière conjointe dans un univers abstrait (formel ou utilisant les notations mathématiques habituelles), et un monde plus concret. On met en relation deux mondes fermés, l'un étant en rapport avec l'abstraction (mathématiques), l'autre avec la réalité (le monde physique) :

- 'The Envisioning machine' de Roschelle décrit dans [GREENO 90] : *« Les élèves manipulent des symboles qui ont un comportement physique »*
- 'The Tarski World' rappelle les études logiques de Pedro GOMEZ au Centre Mondial de l'Informatique, monde des fluides, monde des maisons, associés à la logique des propositions : le jeu consiste à interpréter des formules dans un monde représenté concret.

- COINLAND (Coinland et Numberland) : amener l'élève le plus près possible de l'ultime transition papier/crayon. On a ici des micromondes successifs autorisant peu d'opérateurs : la légalité de leur application correspond quasiment aux possibilités physiques, ce qui évite les erreurs et impasses.
- ARITHMEKIT (voir [BROWN 85] liaison avec les blocs de DIENES),
- Algebra Tutor [MCARTHUR & Al. 88],
- LOGO [FEUERZEIG 86] un micromonde graphique (les sacs de billes).

Dans SHOPPING ON MARS [O'SHEA & Al. 88], le langage GADL (Graphical Arithmetic Description Language) a été créé pour permettre à l'élève de communiquer à la machine ce qu'il a fait pour obtenir un résultat (mode de calcul), ou à l'inverse permettre à la machine de présenter un nouvel algorithme. Une interface de manipulation directe est utilisée et tous les pas de la solution de l'élève sont explicitement présents à l'écran (Voir BRIDGE, ALGEBRALAND).

La même équipe développe aussi deux micromondes :

- 'DIENES block' pour faciliter le transfert entre la manipulation des blocs et l'arithmétique papier/crayon.
- 'Shrink-a-Cube', qui doit aider les enfants à développer et comprendre les principes de l'échange et de la position.

Pour la résolution des 'algebra word problems', on retrouve la même approche, présentée par Greeno [GREENO 87] : utiliser des systèmes graphiques pour enseigner quelque chose sur les notations algébriques et les relier à la sémantique des 'word problems'. D'autres environnements de soustraction sont cités dans [PEA 87]

En marge des micromondes, les jeux sont finalement assez proches, un tutorat minimal permettant des activités de découverte guidée.

II.2.a.4 Caractéristiques générales

Ainsi, on peut distinguer :

- l'approche mathématique/informatique qui en partant du contenu, s'est petit à petit intéressée aux experts puis aux apprenants,
- l'approche diagnostic qui, en partant des apprenants, intègre au fur et à mesure les techniques de l'IA,
- l'approche micromonde qui prône l'adaptation d'outils aux apprenants, et se centre sur l'activité de ces derniers.

On peut dégager une problématique commune à ces systèmes : expliciter comment un mathématicien et un novice s'y prennent pour effectuer un calcul, concevoir une démonstration ou plus généralement résoudre un problème. En particulier, quelles sont les connaissances utilisées, comment choisir un fil conducteur (un plan), quand abandonner une voie qui semble vouée à l'échec, etc. On tente ainsi de faire des résolveurs qui sont proches du mathématicien tout en tenant compte des difficultés du novice, et des connaissances implicites ('tacit knowledge').

Diverses classifications ont été proposées dans l'utilisation de l'ordinateur dans des activités mathématiques de formation, intégrant les approches précédentes et la tradition EAO [HOWE 86] :

- programmes d'application :
  - tableau noir électronique (e. g. Imagiciels [CHASTENET & HOCQUENGHEM 81]),
  - calculatrice électronique,
- programmes d'exercices pratiques,
- programme tuteur,
- programmes de simulation,
- modélisation informatique,
- programmes d'exploration.

Roy PEA [PEA 87] effectue une classification des outils à partir de l'objectif de formation :

- facilitation conceptuelle ('conceptual fluency') : décharger l'élève de certains aspects calculatoires ennuyeux pour se concentrer sur le choix des méthodes,
- exploration mathématique (découverte guidée),
- outils de représentation : maîtriser des représentations multiples et pouvoir passer de l'une à l'autre,
- apprendre à apprendre : savoir évaluer des solutions partielles, contrôle des connaissances stratégiques et de activités de résolution (réification),
- apprendre des méthodes de résolution de problèmes.

Dans la plupart des cas, l'apprentissage est basé sur la résolution de problèmes, l'ordinateur fournissant divers outils de traitement et d'aide sur l'ensemble du processus ou sur certaines étapes. L'ensemble du continuum entre les tuteurs et les environnements d'apprentissage est balayé, et les divers paradigmes d'interaction élève/machine : dialogue socratique, 'learning companion', 'collaborative', 'apprenticeship', etc.

Dans la réalité scolaire quotidienne, l'utilisation des outils informatiques reste encore très embryonnaire. Les enseignants ne sont pas très convaincus et les structures s'y prêtent mal. Les programmes qui se présentent comme des outils pénètrent lentement :

- dans le cadre du cours magistral pour l'enseignant (imagiciels par exemple),
- dans le cadre des séances de travaux pratiques : les activités de découverte sont difficiles à mettre en oeuvre, on a plutôt des progressions rigides qui suivent le modèle des vieux TP de physique. Les outils ouverts sont détournés dans des séances cadrées sans surprises.

Les machines sont rarement accessibles pour permettre des activités aux élèves en dehors du cadre précédent (peu d'utilisation en libre service dans un centre documentaire par exemple).

Enfin, il faut signaler l'utilisation pour l'enseignement de divers outils non directement liés à l'IA :

- les tableurs [HOWE 87],
- les grapheurs et les utilitaires de construction de courbes.

On peut regretter l'absence d'outils intelligents de 'drill and practice' capables d'entraîner les élèves d'une manière autonome sur des activités calculatoires (des exercices nombreux et bien classifiés, des reconnaissances des pistes suivies par les élèves pour des corrections en situation, etc.). Ce type de programme devrait prendre une place importante dans les années à venir, comme complément de l'enseignement traditionnel. En particulier, cela pourrait, dans de nombreux domaines, supprimer l'une des faiblesses majeures de l'enseignement traditionnel qui est l'écart entre la production par l'élève et la correction de son travail.

II.2.b EIAO et contenus mathématiques

Le tableau ci-dessous essaye de fournir une vue synoptique des divers systèmes développés ou en cours de développement. L'ordre choisi suit grossièrement le programme scolaire, du primaire jusqu'à l'université.

	Résolveurs	Diagnostiqueurs	Résolveurs/tuteurs
Opérations + -	SIERRA (*) ACM - DPF	BUGGY, DEBUGGY IDEBUGGY (Young-O'Shea)	SUMMIT PCMATH (Attisha, Yazdani) ARITHMEKIT
/			
Calculs Arithmétique			WEST SHOPPING ON MARS COINLAND
Calcul mental			AMALIA
Décimaux			
Equations 1er degré	ALEX (*)	LMS - PIXIE	ALGEBRALAND EMMA ALGEBRA Alg. Tutor (RAND) Alg. Tuteur (CMI) NAIADE LOGO (Feurzeig) TEACHER'S APPRENTICE etc.
Fractions		BADAUD-FRACT (Visetti)	(Nwana & Coxhead) (Kondo & Al.)
WORD problems	STUDENT		STUDENT PRECEPTOR TAPS, TRIP Tools (LeBlanc) Alg. Word Pb. Tutor (Nathan & Al.) etc.

Factorisation polynomes			APLUSIX NAIADE
Triangles (congruence)	(Gelertner) (Gilmore)		GEOMETRY TUTOR (Greve)
Géométrie Démonstrations Constructions	(Buthion) (Chou)	MENTONIEZH	MENTONIEZH TRICON ARCHIMEDE CABRI
Logique Ensembles	(Wang) (Pitrat) PROLOG LOGIC THEORIST DATTE (Mérialdo)		EXCHECK PLATO EPIC ARRIA MICRO-SEARCH
Calcul Formel Dérivées Limites Combinaisons	CAMELIA (Bonnet) DALI SEME		MACSYMA ADVISOR MATHPERT
Equations d°2 trigo. fract. rat. général	 PRET PRESS, LP (*)		EDUSYM QUADRATIC TUTOR QUADRATIC GRAPHER DISSOLVE
Fonctions			SEDAF
Primitives	SAINT SIN SYCOPHANTE LEX LEX2 (*) METALEX PET (*) TANGO		INTEGRATE INTEGRATION-KID ELISE SIM INTEGRAL
Arithmétique	PARI		
Groupes	(Gillet)		
Inégalités	(Bledsoe)		
Topologie	PROVER MUSCADET		
Combinatoire	ALICE		(Koegel & Al.)
Graphes	GT (**)		
Probabilités Statistiques		PROBIT	(Bergeron & Al.) SAGESSE
"Découverte"	AM (**)		

(*) Systèmes d'apprentissage
(**) Systèmes de découverte

(Ce tableau n'a aucune prétention à l'exhaustivité mais essaye de recenser la plupart des programmes pour mettre en évidence les grandes tendances dans le domaine EIAO et mathé-

matiques)

Nous avons séparé les systèmes en 3 groupes :

1. Ceux dont le but se rapproche plutôt de la démonstration automatique de théorèmes ou du calcul symbolique mais n'ayant pas de prétention primordiale à l'enseignement (résolveurs);
2. Ceux qui sont centrés sur le diagnostic et qui tentent de trouver les erreurs faites par des élèves (diagnostiqueurs);
3. Ceux qui sont conçus pour être utilisés dans un cadre d'enseignement ou explorer diverses stratégies tutorielles (résolveurs/tuteurs).

La lecture de ce tableau amène quelques réflexions :

- les systèmes de diagnostic sont consacrés à des domaines très élémentaires, essentiellement la soustraction, les équations du premier degré (correspondant à la transition difficile entre l'arithmétique et l'algèbre), et les fractions.
- certains domaines sont fortement étudiés :
  - la soustraction,
  - les équations du 1er degré,
  - les 'word problems' (peu de recherches en France),
  - la géométrie : les congruences des triangles dans les pays anglo-saxons, sinon c'est un domaine quasi exclusivement français,
  - la logique,
  - l'intégration formelle.
- les résolveurs couvrent assez bien l'ensemble des domaines (ce qui a permis de valider une certaine universalité de l'approche systèmes experts).

II.2.b.1 Le niveau primaire :

Les études sur le diagnostic prédominent (voir V.1, page 147). On peut noter cependant que les recherches sur les erreurs systématiques conduisant à des classifications plus ou moins détaillées ne débouchent pas sur la réalisation de tuteurs intelligents. Les critiques adressées par R. I. Nicolson à la démarche suivie par BUGGY sont caractéristiques [NICOLSON 88] :

- manque de connaissance pédagogique appropriée : on ne sait pas comment corriger, doit-on associer des remédiations particulières aux déviations constatées (erreurs de surface vs erreurs profondes, V.1.d, page 159),
- trop grand nombre de bugs (330 pour la soustraction!),
- pas de feedback immédiat (un traitement rétrospectif empêche d'effectuer des interventions 'en-ligne', voir la remarque identique d'Odile Paliès [PALIES 88a]),
- instabilité des bugs ('Repair Theory' [BROWN & VANLEHN 80] n'a aucune retombée pratique directe),
- dépendance du médium : les études ont été faites au Nicaragua à partir d'opérations papier-crayon. D'autres environnements de travail fournissent d'autres déviations, ainsi que des méthodes d'enseignement différentes.

- problème pragmatique (écrit en InterLISP, ne peut tourner que sur des machines puissantes inaccessibles aux écoles),
- objections doctrinaires : Nicolson souligne qu'il y a une forme d'anti-arithmétique ambiante qui n'encourage pas le développement de programmes d'enseignement sur un tel thème.

Son système SUMMIT, extension du programme SUMS, intégre uniquement les erreurs les plus fréquentes : pour couvrir 90% des erreurs faites, il suffit de considérer 5 'bugs' pour l'addition et 5 pour la multiplication. Des remédiations spécifiques sont associées à chacun de ces bugs, les interventions étant immédiates en cas d'erreur. A partir de cette classification plus grossière, il constate des phénomènes de stabilité des 'bugs' (voir la simplification des fractions, V.2.b.2, page 164).

On peut cependant remarquer que les chercheurs qui ont travaillé sur les erreurs, ne sont pas partis de ces études pour développer des tuteurs, mais ont adopté une approche de type micromonde (ou le paradigme des deux mondes, II.2.a.3) : ARITMEKIT [BROWN 85], COINLAND [HAMBURGER & LODGHER 89], SHRINK A CUBE [O'SHEA & Al. 88], SHOPPING ON MARS [HENNESSY & Al. 89]. Dans ce dernier cas, le logiciel essaye de reconstituer les méthodes informelles de calcul utilisées par les enfants (étudier les sources d'erreurs, ce qui rejoint les études des didacticiens) et se place dans un contexte de jeu d'aventure. On peut citer aussi [ATTISHA & YAZDANI 83] et PCMATH [AL-KADURIE 88].

Les didacticiens se préoccupent souvent plus des approches des techniques opératoires, dans une perspective voisine des micromondes dans une optique de conservation du sens. Citons le travail de D. Butlen [BUTLEN 85] sur la multiplication, et l'approche de la division [DERAMECOURT & Al. 88]. Il y a d'ailleurs peu de réalisations informatiques sur les problèmes de la maîtrise de la technique opératoire de la division.

Citons enfin, WEST [BROWN & BURTON 82] sur les calculs arithmétiques parenthésés (un des seuls tuteurs en mathématiques ayant conduit à une évaluation externe), et SIERRA [VANLEHN 87] sur la modélisation dynamique de l'élève. L'étude réalisée avec AMALIA sur l'apprentissage du calcul mental est avant tout un test sur l'architecture et l'écriture de plans (rapport interne LIUM).

II.2.b.2 Equations du premier degré

Les travaux sur la résolution des équations du premier degré s'appuient sur les études sur les erreurs et les modes de résolution des enfants (voir [MATZ 82], LMS [SLEEMAN 82], PIXIE [SLEEMAN 83]). Les réalisations sont assez distinctes : des tuteurs, des micromondes et des formes mixtes :

- ALGEBRA [LANZ & Al. 83],
- Tuteur algèbre [FERRET & JIMENEZ 87],
- TEACHER'S APPRENTICE [LEWIS & Al. 87], (relié aux études de Matz complétées par des études sur la factorisation de Siegler et McGilly),
- ALGEBRALAND (FOSS) [COLLINS & BROWN 88],
- Algebra Tutor [MCARTHUR & Al. 88],
- EMMA [QUIGLEY 89],

- NAIADE [JOAB 88], [JOAB & Al. 89] [JOAB 89] [JOAB & Al. 89].

Feurzeig [FEURZEIG 87] utilise LOGO dans un cours d'initiation à l'algèbre comprenant des projets de programmation en LOGO, un micromonde graphique (les sacs de billes), un banc d'essai d'algèbre qui peut se comporter soit en esclave, en exécutant sur commande des opérations algébriques, soit en tuteur donnant avis et critiques.

NAIADE tente de s'attaquer à trois aspects principaux du processus cognitif :

- les novices travaillent sur des théories locales sans lien entre elles,
- les connaissances sont liées à des contextes spécifiques. Les stratégies utilisent des similarités de type perceptif, plus que fonctionnel ou conceptuel (voir V.3, page 171 et VI.3.b, page 194),
- le travail est local, il n'y a ni plan ni de stratégie globale.

ALEX (Algebra Learning from EXamples) fut conçu par D. M. Neves pour apprendre à résoudre des équations du premier degré à partir d'exemples tirés de manuels (cité par R. H. Wenger [WENGER 87]).

II.2.b.3 Les 'word problems'

L'ancêtre est le système STUDENT [BOBROW 64]. L'accent est souvent mis sur le problème de la représentation, avec utilisation de techniques d'animation ou de dessin. L'animation est soit vérifiée complètement ou partiellement par le système, soit laissée à la sagacité de l'élève, et peut être reliée aux équations. Des outils assistent l'élève dans la phase de résolution des équations.

- TRIP [GOULD & FINZER 81] écrit en Smalltalk, permet la gestion d'une animation. L'équation doit être spécifiée par l'enseignant, et divers paramètres sont ajustables. Le système peut juger la figure réalisée par l'élève en tenant compte de 50 contraintes. Les auteurs ont pu observer que les élèves ne savent pas où regarder pendant une séquence animée, ce qui ne facilite pas la phase de formalisation.
- PRECEPTOR [SCHULZE 89] traite de problèmes de monnaie, de pièces, et de calcul de périmètre. On arrive à des équations avec 2 inconnues (parfois à valeurs entières). Le mode est tutoriel sur l'écriture de ces 2 équations avec différentes stratégies adaptées à l'historique de l'élève.
- TAPS [DERRY & Al. 88] [DERRY & Al. 89] (voir V.1.c.5, page 157).
- [LEBLANC 88] boîte à outils pour aider l'élève à traduire l'énoncé : faire des dessins, trouver des relations, organiser les données, expérimenter.
- [NATHAN & Al. 89] : utilisation d'un programme d'animation (ANIMATE).
- [HALL 89] : utilisation de diagrammes qualitatifs.
- The Algebra Word Problem Tutor [SINGLEY & Al. 89] fournit des outils, une aide corrective et du feedback sur les erreurs.

Dans ces divers systèmes, l'idée est souvent de fournir une représentation de la structure des informations (Voir [GREENO 87]) : comprendre un problème nécessite d'identifier les quantités ainsi que les relations entre elles.

II.2.b.4 Les fractions

Une étude détaillée sur le diagnostic d'erreurs dans le calcul des fractions sera vue plus loin (voir V.2.b, page 162). Divers systèmes incluent une perspective diagnostique dans une stratégie tutorielle :

- [VISETTI 86] (V.1.c.4, page 157),
- [NWANA & COXHEAD 88],
- [KONDO & Al. 90].

Des réalisations existent dans un cadre de formation professionnelle [MERRI 90] en liaison avec la proportionnalité (voir aussi un travail mené en Logo [HOYLES & Al. 89]). D'autres études se focalisent plus sur les problèmes de représentation : [OHLSSON 87], [ZUIDENA 90].

II.2.b.5 Résolution d'équations et d'inéquations

Divers systèmes ont été réalisés, résolvant divers types d'équations spécifiques ou généraux :

- QUADRATIC TUTOR [O'SHEA 82], équations du second degré;
- QUADRATIC GRAPHER [LOSER & KURTZ 89], inclut les représentations graphiques;
- EDUSYM [JURKOVIC 87] équations polynomiales,
- DISSOLVE [OLIVER & ZUKERMAN 90] fractions rationnelles,
- SEDAF [AIELLO & Al. 88] sur les études de fonctions rationnelles.
- PRESS [BUNDY 79], [BUNDY & WELHAM 81] résolveur introduit dans le programme MECHO [BUNDY & Al. 79], LP [SILVER 86].

La méthode de résolution de PRESS est basée essentiellement sur l'isolation (application de fonctions réciproques), le rassemblement (diminuer le nombre d'occurrences de l'inconnue) et l'attraction (rapprocher deux occurrences de l'inconnue), l'homogénéisation (préparatoire aux changements de variables), l'élimination de symboles fonctionnels rares et la reconnaissance de formes particulières.

Certains systèmes traitent de manière assez étendue des problèmes de factorisation : NAIADE [JOAB & Al. 89] et surtout APLUSIX [NICAUD 87a] qui est en cours de test dans des écoles.

Dans les résolveurs, on peut citer rapidement :

- PRET [GRANDBASTIEN 74], résolution d'équations trigonométriques à l'aide de procédés heuristiques,
- Calcul des dérivées [BONNET & Al. 81],
- SEME [BARON 82], évaluation de formules sommatoires en analyse combinatoire,
- Inégalités dans R [BLEDSOE & Al. 85] (pas de tuteur encore sur ce thème).

Enfin des systèmes qui traitent de manière générique du calcul formel :

- MACSYMA ADVISOR [GENESERETH 82], assistance à l'utilisation de MACSYMA,
- CAMELIA [VIVET 84],
- MATHPERT [BEESON 89].

II.2.b.6 La géométrie

La géométrie a été l'un des premiers domaines auquel la démonstration automatique de théorèmes s'est intéressé : [GELERTNER 63], [GILMORE 70], [CHOU 88], [BUTHION 75] (construction de figures géométriques).

Le système le plus connu est GEOMETRY TUTOR [ANDERSON 83] [ANDERSON & Al. 85]. Il commence à être utilisé dans les écoles [SCHOFIELD & Al. 90]. Sur le même domaine de la congruence des triangles, un système est développé par Grève [GREVE 89].

En France, le domaine de la géométrie intéresse de nombreux chercheurs. Des articles généraux ([GUIN 89] [CUPPENS 90]) dressent un panorama assez complet des réalisations actuelles et en cours.

On peut citer brièvement :

- ARCHIMEDE [CHOURAQUI & INGHILTERRA 87], qui s'intéresse à la mise au point d'une base de connaissances en géométrie et au raisonnement par analogie,
- MENTONIEZH [PY 89], démonstrateur en géométrie et tentative de détermination des intentions d'un élève à partir de ses productions,
- CABRI [BAULLAC 90], CAhier de BRouillon Informatique, essentiellement un micromonde pour les constructions géométriques. Une extension vers un CABRI intelligent est en cours de réalisation.

Enfin, il faut signaler, en Allemagne, le système TRICON [HOLLAND 88].

II.2.b.7 Logique et théorie des ensembles

De nombreux résolveurs ont été réalisés : [WANG 60], DATTE [PASTRE 76], Mérialdo [MERIALDO 79], etc.

Citons rapidement :

- EXCHECK [McDONALD 81] : système utilisé régulièrement en formation à l'université de Stanford,
- MICRO-SEARCH [SLEEMAN 87a] : montre l'unification en mettant les expressions concernées en surbrillance,
- PLATO [WALSH 88],
- EPIC [TWIDALE 89] (voir V.1.c.6, page 158).

II.2.b.8 Le calcul de primitives

L'intégration formelle est un peu le domaine d'expérimentation privilégié de nombreuses recherches en Intelligence Artificielle. Tout d'abord dans l'implantation de méthodes de résolution heuristiques :

- SAINT [SLAGLE 61]
- SIN [MOSES 71a] utilisé dans MACSYMA,

- SYCOPHANTE [BERGMAN & KANOUI 78] système de calcul formel plutôt spécialisé dans la recherche d'intégrales (les thèses de BERGMAN et KANOUI sont sur ce thème),
- TANGO [ROUSSET 83],
- CAMELIA [VIVET 84].

Ensuite, dans les recherches sur l'apprentissage automatique :

- LEX [MITCHELL & Al. 83], puis LEX2,
- METALEX [KELLER 87],
- PET [PORTER & KIBLER 86].

A ce titre, on peut se demander si ce domaine est réellement intéressant en EIAO ou si les travaux qu'il suscite ne sont pas dus à des déformations des recherches fondamentales en IA. En particulier, les connaissances mathématiques interviennent relativement peu, il s'agit plus d'un ensemble organisé de recettes qu'on essaye d'appliquer en espérant que le cas traité ne leur résiste pas (ce que l'on peut difficilement savoir à l'avance, sauf si les exercices ont été consciencieusement sélectionnés).

On peut mentionner :

- INTEGRATE [KIMBALL 82] où on constate que les étudiants trouvent rapidement de meilleures solutions que le système!
- SIM [BEAULIEU 88] qui s'appuie sur les travaux de [SCHOENFELD 85] qui a élaboré une stratégie dite de contrôle en s'inspirant des techniques de l'IA suite à l'observation d'experts lors du processus de résolution de problèmes. Les étudiants ne sont pas capables de développer des stratégies efficaces de contrôle. Une stratégie prescriptive influence leur performance.
- INTEGRATION-KID [CHAN & BASKIN 88] qui constitue plutôt un test d'une architecture particulière (LCS : 'learning companion system').
- ELISE [CARRIERE & DELOZANNE 89] qui se base sur l'analyse effectuée par Rogalski.

Le logiciel INTEGRAL de Fortin (IREM de Rouen) bâti au-dessus de Mu-Math est distribué.

II.2.b.9 Autres

Divers résolveurs abordent d'autres domaines des mathématiques :

- combinatoire : ALICE [LAURIERE 76],
- arithmétique : PARI [BOURGOIN 79],
- théorie des groupes [GILLET 79],
- espaces vectoriels topologiques : MUSCADET [PASTRE 84].

A la suite de AM [DAVIES & LENAT 82], [LENAT 83], les chercheurs tentent de concevoir des systèmes capables de faire des découvertes et s'orientent dans des domaines très avancés des mathématiques comme la théorie des graphes (GT [EPSTEIN 87]), voir [LAUBLET 87].

II.2.c Problématiques de développement

II.2.c.1 Outils, Modèles élèves, interactions :

En ce qui concerne les outils généraux, on peut lister :

- les systèmes de calcul formels [ASPETSBERGER & KUTZLER 88], [LAURENT-GENOUX & Al. 88]
- les langages de l'IA : LOGO, LISP, SMALLTALK, PROLOG.
- les systèmes de construction géométrique.

Les systèmes basés sur le diagnostic des erreurs ('buggy model') sont plutôt de type tutoriel, puisqu'il n'y a pas de moyen simple pour que l'élève se rende compte par lui-même qu'il se trompe. Le système doit donc prendre l'initiative et intervenir, ce qui conduit à la remédiation. Il faut cependant rappeler que l'équipe de BUGGY n'a pas développé de tuteur sur la soustraction ou l'arithmétique élémentaire, utilisant plutôt les concepts dégagés dans la modélisation dans des projets de physique (SOPHIE).

Une approche avec des outils et des opérateurs introduit des modèles de type expertise partielle ('overlay model'), proche des techniques de WEST : associer des descripteurs aux opérateurs disponibles précisant le type de maîtrise de cet opérateur par l'utilisateur. C'est le cas de MATHPERT [BEESON 89]. Ceci se combine avec un mode d'interaction articulé sur des choix d'opérateurs dans des menus, les expressions successives étant présentes à l'écran sous une forme linéaire ou arborescente (ALGEBRALAND ou GEOMETRY TUTOR). La granularité des transformations choisies peut être adaptée, les étapes étant plus ou moins détaillées (APLUSIX, Tuteur algèbre).

Au niveau de l'interaction, on oppose aussi les interfaces de manipulation directe (CABRI) aux langages de commande (EUCLIDE). Il faut remarquer que ces deux approches sont complémentaires : la première est plus efficace au niveau de la construction des objets, mais peut manquer de représentation alternative sur laquelle travailler (difficulté de restitution d'états antérieurs ou de création de nouveaux objets).

L'un des aspects les plus centraux est la possibilité de réification : [LESGOLD 87] « *On peut penser que le rappel concret de ce qu'a fait l'élève peut l'aider dans ce processus (construction consciente de son savoir). Ce qui manque jusqu'à présent, c'est la rétroaction sur le savoir que l'élève vient de construire. Dans une certaine mesure, l'ordinateur pourrait les déduire de la structure de la représentation de la même manière que nous demandons aux élèves de procéder à ce type d'inférences.* »

II.2.c.2 Quelques questions

L'EIAO dans le champ des mathématiques pose un certain nombre de questions quelles que soient les approches poursuivies. On peut en lister quelques unes :

1. Comment naissent les erreurs au cours de l'apprentissage, peut-on et faut-il éviter leur émergence? (voir V.3.a, page 171)
2. Existe-t-il des remédiations efficaces et faut-il les associer à des bugs ou des fausses règles ou à des conceptions erronées plus profondes? (voir V.3.b, page 173)

3. De quelle façon intégrer les divers résolveurs dans l'enseignement et quelles modifications risquent-ils d'introduire dans les programmes?

   " What forms of algebraic skill or knowledge are needed to use powerful symbolic algebra environments with understanding? ... What methodologies can be used to identify those forms of algebraic understanding that will be important in the future. " [WENGER 87]

4. S'il faut travailler sur de multiples représentations, lesquelles choisir ou inventer? faut-il remettre en cause les notations mathématiques usuelles?

5. Faut-il compléter les programmes en ajoutant l'enseignement des heuristiques de résolution liées aux domaines mathématiques abordés, est-ce véritablement un objet d'enseignement?

   Il y a déjà le premier problème de la formulation de toutes ces heuristiques. Il faut bien noter que l'on retrouve le problème général de la constitution des bases de connaissance dans les systèmes experts. Trouver des heuristiques de qualité est très difficile et nécessite une grande pratique du domaine. Il est peu probable que les enseignants puissent effectuer ce travail qui ne peut être confié qu'à des experts. Ces nouvelles connaissances risquent de ne pas être connues des enseignants qui peuvent les rejeter ou mal les intégrer (voir ARRIA IV.2, page 127, CAMELEON VI.4, page 199).

6. Peut-on enseigner des habiletés générales de résolution de problèmes?

Chapitre II.3

Impact sur l'enseignement

> « *La plupart des interventions éducatives ont pour ambition de produire des effets à long terme plutôt que des effets à court terme. Cependant, les recherches en évaluation sont tellement dominées par une conception mécaniciste ... que nous avons appris à faire comme si un résultat, de quelque nature qu'il soit, qui montre un effet immédiat, constituait l'impact essentiel d'un programme d'éducation. Une telle conception aboutit à une approche superficielle et réductrice des problèmes en éducation.* »
>
> *[ZIMILES 77]*

L'étude de l'impact de l'EIAO sur l'enseignement concerne en priorité l'utilisation des outils développés, ainsi que leur expérimentation et évaluation, en particulier pour vérifier leurs prétentions vis-à-vis de la formation et les comparer avec d'autres modes d'enseignement. On peut ainsi noter que l'évaluation formelle des grands projets américains PLATO et TICCIT n'a pas été très concluante (voir [O'SHEA & SELF 83]). Mais, au-delà de l'utilisation de produits développés, les diverses études menées permettent de mieux comprendre les processus d'apprentissage ce qui peut avoir des retombées sur l'enseignement dit traditionnel.

II.3.a Evaluation / Expérimentation

II.3.a.1 Faut-il expérimenter et évaluer?

Cette question de la nécessité ou non d'une évaluation conditionne l'approche que l'on a du domaine de l'EIAO. On peut considérer que l'EIAO concerne avant tout la recherche fondamentale, qu'elle travaille sur le long terme et qu'elle fournit plutôt des modèles que des produits.

" *The demand for empirical evaluations in realistic settings may have blinded ITS designers to the view that, in the longer term, these are precisely what they should be designing to avoid.*

The present experimental strategy has (in apparent paradox) led to the ITS field acquiring an unenviable reputation for the non-delivery of 'working systems'. " [SELF 89]

Une position diamétralement opposée part de la constatation que la plupart des systèmes ont été développés comme outils de laboratoire pour tester des hypothèses relatives à des aspects spécifiques de l'apprentissage avec les ITS. Il faut maintenant que les recherches de base soient confrontées aux contraintes et aux problèmes de réels environnements d'apprentissage :

- formation d'adultes,
- entraînement technique (dépannage, diagnostic [MOUSTAFIADES 90]),
- formation sur le site ou sur le lieu de travail.

Johnson [JOHNSON 86] remarque que si les aides intelligentes et les ITS sont développés indépendamment, il est vraisemblable que des fonctions seront perdues ou dupliquées dans leur intégration. Cela pourrait cependant servir à redéfinir le concept d'entraînement d'apprentis ('apprenticeship training').

L'expérimentation est un processus en spirale vers l'inconnu : on découvre de nouveaux buts et de nouvelles idées, etc. Elle fait partie intégrante de la recherche et ne peut en être séparée. C'est une des idées centrales de la présente thèse (voir expérimentation).

Remarque : l'un des seuls logiciels référencés dans le domaine de l'EIAO (même s'il n'en possède pas toutes les caractéristiques) est EXCHEK [McDONALD 81] qui traite de la construction de preuves en théorie axiomatique des ensembles.

II.3.a.2 Comment évaluer?

Même si on reconnaît que l'expérimentation est nécessaire, évaluer correctement ses résultats bute sur la complexité des phénomènes d'apprentissage. On laissera de côté les classiques évaluation sommative et évaluation formative, qui font partie intégrante du folklore de la formation et ne sont pas pertinentes dans ce contexte (sauf dans le cadre de l'utilisation des techniques d'apprentissage automatique comme le fait K. Van Lehn [VANLEHN 90], mais qui ne vise pas directement la réalisation de logiciels d'enseignement).

L'évaluation peut être assurée de deux façons :

- par le jugement d'un professionnel ou d'un expert,
- par l'accumulation de données concernant l'adéquation des performances aux objectifs.

La plupart des évaluations qui ont été effectuées sont des évaluations subjectives du premier type. Il est en effet difficile de préciser des objectifs et la situation évolue rapidement. De plus, les apprentissages nécessitent un temps long qui est rarement compatible avec les durées effectives des recherches.

D. Littman et E. Soloway [LITTMAN & SOLOWAY 88] préconisent deux types d'évaluation, qu'ils ont utilisé pour évaluer le tuteur PROUST sur l'apprentissage du langage PASCAL [JOHNSON 86] :

- une perspective cognitive : évaluation externe basée sur la construction d'un modèle élève puissant pour isoler certains aspects du système devant avoir un effet particulier sur le processus d'apprentissage des élèves. L'idée est de savoir comment les élèves

résolvent les problèmes et non pas s'ils peuvent les résoudre. Ainsi, l'évaluation d'un tuteur intelligent peut être profondément différente de l'évaluation d'un logiciel d'EAO. Cette dernière se centre prioritairement sur les réponses correctes et incorrectes. La première s'appuie sur les raisons pour lesquelles les élèves donnent des réponses correctes et incorrectes. Ceci n'est cependant pas sans difficultés pratiques, puisqu'il faut trouver ce modèle élève (V.1.b, page 152). Littman et Soloway centrent leur évaluation externe de PROUST sur des aptitudes de déboguage de programmes.

- une perspective sur l'architecture : relations entre l'architecture d'un ITS et son comportement, correspondant aux réponses aux trois questions suivantes :
  - Que connaît l'ITS?
  - Comment l'ITS fait-il ce qu'il fait?
  - Que devrait-il faire?

  Mener une telle analyse interne engendre souvent la réalisation de versions successives, s'appuyant sur les erreurs constatées. Son intérêt réside aussi dans le fait d'éviter le recours obligatoire à une évaluation externe formelle. En d'autres termes, la structure même d'un ITS fournit de nombreux renseignements sur son efficacité.

Les évaluations externes formelles sont plutôt rares. Littman et Soloway citent PROUST et LISP TUTOR [ANDERSON & REISER 85]. Dans les deux cas, il s'agit de l'apprentissage de langages de programmation à l'université : l'évaluation est facilitée parce qu'elle peut être effectuée dans le cadre de travail des développeurs. Une exception concerne WEST [BURTON & BROWN 82] jeu repris de PLATO.

Des rapports sur l'usage de GEOMETRY TUTOR [ANDERSON & Al. 85] dans une classe ([SCHOFIELD & EVANS-RHODES 89], [SCHOFIELD & Al. 90]) sont plus concernés par son impact sur la structure et le fonctionnement de la classe qui constitue une évaluation interne de Geometry Tutor.

Les ITS étant trop complexes, on est contraint à un mode de production cyclique : production, évaluation faiblesses et forces, nouvelle version, etc. L'évaluation peut même être faite avec des ITS partiellement développés et des modèles élèves incomplets. On peut regretter ce que l'on peut considérer comme un aspect encore artisanal, mais en l'absence d'une 'théorie de l'éducation' précise bien acceptée pour guider la création des programmes, ces derniers sont écrits avec des spécifications pauvres. Ils sont testés avec les enseignants et les élèves et sont ensuite modifiés suivant leur réaction.

" Testing then becomes a crucial, constructive stage of the design process. " [SELF 85a]

II.3.a.3 Evaluation de LOGO

« Ne demandez pas ce que LOGO peut faire pour les gens, mais ce que les gens peuvent faire avec LOGO » [PAPERT 85]

Les modes d'évaluation associés à une telle prise de position sont fortement subjectifs : prendre un groupe d'étudiants, travailler assez longtemps avec eux pour tester l'innovation et espérer des résultats qualitativement différents. A la limite, on peut observer un seul individu (Lawler et sa fille [LAWLER 81] ou le travail de L. RESNICK).

Les expérimentations autour de l'usage de Logo n'ont jamais donné de très bons résultats. Ainsi, dans le domaine mathématique, les élèves ayant travaillé avec Logo n'ont pas de

meilleures performances que les autres. Un point positif souligné [HOWE & Al. 80] [ROSS & HOWE 81] cependant est que les enseignants rapportent que les élèves pouvaient :

- argumenter de manière sensée sur des points mathématiques,
- expliquer clairement des difficultés mathématiques.

La majorité des études empiriques se sont attachées à déterminer dans quelle mesure programmer en Logo augmente la capacité des enfants à résoudre des problèmes. Le bilan de ces recherches est décevant [CRAHAY 87]. « *... il ne semble pas raisonnable d'attendre des effets de transfert de l'apprentissage de la programmation sur des compétences et des aptitudes de haut niveau* ». [MENDELSOHN 88] Ce n'est pas très étonnant, puisque « *les psychologues qui se sont penchés sur les mécanismes de résolution de problèmes ne cessent de répéter qu'il n'est pas possible d'identifier un ensemble d'opérations mentales ou de capacités spécifiques qui soient efficaces dans tous les cas... l'hypothèse des structures d'ensemble ne correspond pas aux données empiriques recueillies ces dernières années; de même on abandonne l'hypothèse de procédures de traitement de l'information opérant en toutes circonstances.* » [CRAHAY 87]

Marcel Crahay insiste sur les déviations et récupérations du constructivisme sous-jacent à Logo par les doctrines pédagogiques traditionnelles :

- récupération rationaliste (l'idée que « *toute faculté, une fois développée à son potentiel maximal, se manifeste en toutes circonstances* »),
- récupération empiriste, où on réduit l'évaluation à l'appréciation de la quantité de connaissances maîtrisées et à une mesure de transfert général,
- récupération libertaire, où sous prétexte de non-directivité, le rôle de l'enseignant est minimisé voire carrément supprimé.

Les recherches suivantes ont ainsi essayé de relier les interventions des enseignants aux apprentissages souhaités. Ainsi [PEA & KURLAND 84], [PEA & Al. 85] précisent que le développement des aptitudes liées à la résolution de problèmes dépend du fait que l'on ait effectivement montré aux enfants comment le langage pouvait être utilisé.

La thèse de McCoy Carver [McCOY CARVER 86] sur les transferts d'activité de déboguage montre que les enfants peuvent apprendre des aptitudes intellectuelles de haut niveau si les habiletés qui les composent sont spécifiées et enseignées directement. Une fois que ces aptitudes ont été apprises, elles peuvent être transférées lorsqu'elles sont reconnues comme se rapportant à la nouvelles tâche. Est exprimé ici le problème de la reconnaissance du contexte.

L'évaluation porte aussi sur le mode de 'tutoring' [SWAN 89]. Les résultats indiquent qu'un enseignement explicite et la pratique médiatisée de la programmation Logo, et seulement de telles formes d'instruction et de pratique, induisent le développement de quatre stratégies particulières de résolution de problèmes : formation de sous-buts, chaînage avant, essais et erreurs systématiques, et analogie. Ces résultats témoignent de la supériorité de l'instruction explicite sur la découverte guidée et des environnements informatiques sur les manipulations concrètes pour l'enseignement et l'apprentissage de telles stratégies.

En résumé, si l'évaluation de Logo a fait couler beaucoup d'encre, il semble que l'intérêt des micro-mondes soit maintenant reconnu, mais qu'il est nécessaire d'y associer un mode de soutien, qui est déterminant dans l'efficacité du dispositif. Ceci pose ainsi le problème de la capacité de l'enseignant à mettre en oeuvre une pédagogie constructiviste à l'aide des ordinateurs. Sous forme de boutade, on a pu dire qu'évaluer Logo revenait plus à évaluer l'enseignant

responsable de la formation en Logo, ce qui est un mode de lecture des expérimentations légèrement différent!

II.3.a.4 Possibilité d'un transfert?

Le problème du transfert est un peu la pierre d'achoppement de la pédagogie. Si on peut définir cette notion de transfert comme « *simplement appliquer des informations d'une catégorie connue à une nouvelle instance* » [McCOY CARVER 86], les chercheurs ont une idée plausible mais vague des similarités entre les tâches source et but, et les moyens de faciliter cette application ne sont pas encore très clairs.

McCoy Carver, dans son rapport déjà cité sur le transfert des aptitudes de déboguage, note une observation assez curieuse (remarquée par F. Robert). Bien que les enfants ont été capables d'utiliser les structures des programmes pour guider le déboguage des programmes des expérimentateurs, ils n'ont pas incorporé de telles structures dans leurs propres programmes. Ainsi, ils ont eu plus de difficultés à déboguer leurs propres programmes que ceux des autres [McCOY CARVER 86]. On peut se demander si un apprentissage réel a pu être effectué, du fait que les élèves n'ont pas l'air d'avoir créé des structures d'une généralité suffisante.

Nous verrons plus loin que l'une des hypothèses est, dans le cadre de la résolution de problèmes, de travailler plus sur la recherche en général que sur l'apprentissage d'heuristiques particulières (voir VI.3, page 193).

Une deuxième idée est de rendre le plus stable possible la forme externe des outils informatiques pour favoriser les reconnaissances de similarité entre différents domaines (voir IV.3.e, page 143).

Une troisième idée consiste à privilégier les modes d'accès aux connaissances plus que les connaissances elles-mêmes, et d'expliciter les liens entre ces connaissances (voir III.1.b, page 85).

Une dernière idée est de revoir les cursus d'enseignement pour éviter la génération de fausses inductions (voir V.3.b.2, page 174).

Dans le problème des élèves et des professeurs (voir page 52), l'une des hypothèses que l'on peut faire, est que les étudiants qui font l'erreur d'inversion n'ont pas l'habitude de vérifier les formules en faisant une ou deux substitutions numériques rapides pour les variables numériques. Il semble raisonnable de penser que la programmation peut avoir ici un effet bénéfique [SOLOWAY & Al. 83]. Cette présente thèse aurait d'ailleurs pu être un travail sur ce thème, i. e. sur les transferts possibles des activités de programmation vers la résolution de 'Word Problems' (Collaboration entre le Centre Mondial de l'Informatique et l'équipe de Soloway, malheureusement, la direction des écoles a refusé son soutien).

II.3.a.5 Les différents acteurs

Dans l'impact des technologies de l'éducation, le rôle des enseignants est essentiel. Ils gardent leur liberté de choix, et déterminent le type d'utilisation réellement fait par les apprenants. Aucun système développé ne peut encore prétendre permettre une auto-formation complète, les actions éducatives associées conditionnent donc les apprentissages. C'est une des conclusions des évaluations de Logo.

L'avis qu'ont les enseignants de l'intérêt des nouvelles technologies donne une mesure de leur impact. Ainsi, dans les grands programmes PLATO et TICCIT, les enseignants étaient principalement concernés par :

- leur propre autonomie
- leur propre interaction avec les élèves [ALDERMAN & MALHER 77].

Une étude internationale récente dresse un premier bilan :
« Les instituteurs et les enseignants de français, sciences et mathématiques qui utilisent l'ordinateur en classe sont d'accord pour observer des changements positifs chez les élèves. Ces changements s'observent surtout dans le domaine de la collaboration entre élèves (79 à 80%), du mode de travail - travail en groupe (59 à 66%) ou individualisé (51 à 63%) - et de l'intérêt pour la discipline (77 à 84%). » [BARRIER 90]
Cette idée de collaboration (voir collaboration) semble donc un point essentiel pour les enseignants : cela favorise le dialogue et les conflits cognitifs entre les élèves.

Cette observation fournit aussi des indications sur le type de logiciels qu'il semble utile d'intégrer à la formation. Au sujet des logiciels d'EAO, [van der VEER & BEISHUIZEN 84] remarquent que dans les écoles primaires, il ne semble pas y avoir de besoin pour ce type de logiciels, les enseignants n'ont pas besoin d'être remplacés. Les logiciels utilisant des techniques de choix multiples ne sont pas applicables, il est préférable de créer des outils qui nécessitent des techniques de 'design' avancées. McKenzie [McKENZIE 90] note aussi qu'en Angleterre, se dessine une nette préférence vers des modèles de type outil plutôt que tutoriel, les nouveaux produits servent plus de complément au processus éducatif que de remplacement des méthodes traditionnelles.

Les enseignants peuvent tout aussi bien faciliter l'accès à ces nouvelles technologies qu'en détourner ou pervertir l'usage. Les compétences générales des enseignants ainsi que leurs connaissances des nouvelles technologies sont déterminantes :

- conception de l'informatique et de son usage en éducation : le modèle classique de l'E. A. O. est souvent dominant, et constitue un filtre dans l'abord des logiciels pédagogiques. Les détournements de LOGO sont fréquents, où on privilégie le contrôle au détriment de la résolution (faire afficher des figures et demander leur nom à l'enfant).
- représentation des outils : les enseignants sont des utilisateurs et passent par des phases de sous-estimation / surestimation des logiciels qu'ils utilisent. Ce phénomène s'amplifie avec les outils montrant quelques lueurs d'intelligence (ils tendent en effet à induire des attentes irréalistes de la part des utilisateurs naïfs [WENGER 87]).
- connaissances sur le domaine : l'expertise du domaine implantée dans les outils informatiques est souvent différente des connaissances habituelles. Les décalages amènent diverses difficultés.
- modes d'usage : ils sont en majeure partie à découvrir. Il faut pouvoir inclure de la documentation pour les élèves, mais aussi pour les enseignants.

Au-delà des enseignants, il ne faut pas mésestimer l'importance de tout l'environnement qui exerce des pressions plus ou moins directes et plus ou moins fortes sur les activités scolaires : les décideurs, les formateurs, les évaluateurs, etc. Leur prosélytisme ou leur blocage vis-à-vis des nouvelles technologies a des effets directs sur l'appropriation qui peut en être faite par les enseignants.

On peut remarquer que si l'enseignement traditionnel n'est pas prêt à intégrer de manière significative les nouvelles technologies éducatives, l'impact sur les enseignements spéciaux (handicapés, enfants malades ou isolés, dans un cadre d'enseignement à distance) peut être beaucoup plus important. Ce phénomène devrait en particulier influer sur le type de recherches menées en EIAO, et la prise en compte de leur intégration dans les organisations actuelles.

II.3.b Autres retombées de l'EIAO

En dehors de leur usage premier, théoriquement la formation des apprenants, les systèmes d'enseignement intelligents peuvent avoir d'autres impacts :

- formation initiale et continuée des maîtres [CUPPENS 88],
- refonte de l'enseignement dispensé : « *Les nouvelles technologies de l'information remettent en question le programme d'enseignement des compétences de base. Bien plus, elles fournissent l'occasion de repenser l'enseignement.* » [LESGOLD 87],
- réflexion des enseignants sur leur style d'interaction avec les enfants et leurs décisions pédagogiques personnelles (effet de bord [LEWIS & Al. 87] de l'émulation d'un style de tuteur, peut être utilisé comme système inductif pour récupérer les décisions, plutôt de type opportunistes),
- réflexion générale sur l'apprentissage,
- changements dans l'organisation (nécessaires pour l'introduction des machines dans les écoles).

Chapitre II.3. Impact sur l'enseignement

Troisième Partie

Hypertexte et EIAO

" It is of course inevitable. It should have been obvious to anyone that general interactive media would appear and proliferate. Text, graphics, audio and video can now come alive in unified, responding, explorable new works that present facts and ideas : hypermedia. Unlimited new forms of connection and branching now offer the chance to explore ideas - to follow different lines of thought, different forms of exposition, different connections in a subject, in ways never before possible. The sequential writings and media of the past have given us only the dimmest precedents. "

(Theodor Holm Nelson [NELSON 89])

" The history of the use of new technologies in education tends to reflect the search for panaceas rather than the serious attempt to solve problems. The great temptation with the adoption of a new medium is to think that it may cause the old problem to disappear. "

[WHALLEY 90]

" While it offers power, this very power ought not feed our intellectual hubris. Our hope for the future of this technology lies in the recognition that there is nothing conceptually neutral in the cognitive universe, certainly not hypertext. "

[DOLAND 89]

Chapitre III.1

Généralités : hypertexte et E. I. A. O.

III.1.a Hypertexte / Hypermedia

III.1.a.1 Notion d'hypertextes et d'hypermedia

III.1.a.1.a Définition

Le terme d'hypertexte désigne des systèmes permettant l'inter-connexion de divers types de documents, non sur un modèle hiérarchique ou relationnel, mais par association d'idées. Leur utilisation correspond à une forme de lecture en profondeur, ce qui explique la dénomination «hyper». Il s'agit d'une nouvelle forme de communication non-linéaire et multi-dimensionelle. En effet, on peut considérer que l'expression orale est essentiellement linéaire, le texte (l'écrit imprimé) est plan, mais inclut des modes d'accès complémentaires (tables des matières, tables d'index, références croisées). L'hypertexte ajoute une nouvelle dimension, les documents de ce type étant destinés à être lus et éventuellement complétés selon différents chemins suivant le choix du lecteur (Voir [CONKLIN 87] pour une présentation générale) : un bon hypertexte est un medium qualitativement différent, il donne à son lecteur le sentiment de se mouvoir sans effort à travers un environnement d'informations transparent, comme un poisson dans un océan de connaissances [DAVIS 90] (voir en contrepoint III.3.a.1, page 103).

Le terme d'hypermedia désigne en général un hypertexte incluant des ressources multimedia, i. e., en plus du texte, des images, des sons, des séquences vidéo, etc. Dans la suite, on gardera le terme générique hypertexte pour désigner aussi bien des systèmes limités aux ressources textes qu'aux hypermedia.

Cette notion d'hypertexte, qui correspond d'une manière assez générale à la possibilité d'établir ou de naviguer à travers des liens multiples entre des documents de natures différentes, recouvre en fait deux idées complémentaires :

- l'idée d'intégrateur, i. e. de système capable de piloter des ressources différentes (produites par d'autres programmes),

- l'idée de système d'organisation des informations qui permet de les stocker et d'y accéder de la même façon que le cerveau humain accède aux siennes, i. e. de manière associative.

L'hypertexte est avant tout un concept, qui conduit à des réalisations très diverses n'incluant pas toujours complètement les deux notions précédentes. L'idée d'intégrateur, en particulier, est encore trop souvent absente ce qui oblige l'utilisateur à quitter son environnement informatique habituel pour utiliser les fonctions hypertextes et hypermedia [MEYROWITZ 89]. Cette composante est indispensable pour faire de l'hypertexte véritablement un outil destiné à améliorer la gestion de l'ensemble des informations et de les rendre effectivement disponibles et accessibles à différents utilisateurs, qu'ils soient experts ou novices.

III.1.a.1.b Structure

Un hypertexte se compose de noeuds et de liens. Les noeuds sont les éléments d'informations, i. e. les documents ou les cartes, l'ensemble des ressources consultables, les liens correspondent aux connections entre les noeuds et sont visualisés sur l'écran par des boutons. Les noeuds ou les liens peuvent être typés. GUIDE, par exemple, utilise 4 types de liens :

- les liens de référence (passage d'un document à un autre),
- les liens de remplacement (internes au document),
- les liens de définition,
- les liens de commande.

L'ensemble constitue un réseau d'idées qui diffère d'une base de données par la structure associative de l'information et le contrôle dynamique offert aux utilisateurs. Les parcours ou chemins dans l'hypertexte peuvent être prévus par l'auteur, construits dynamiquement par l'utilisateur ou provenir de responsabilités partagées. Des «visites guidées» (guided tours) sont souvent incluses, correspondant à des chemins prédéterminés particuliers. Ces coupes sont très utiles pour répondre à des besoins spécifiques répertoriés (première approche d'un domaine, poursuite d'un but déterminé, etc.).

Pour faciliter la navigation dans le réseau, certains systèmes offrent une forme de visualisation du graphe. L'historique de la consultation est généralement conservé, l'utilisateur a la possibilité de revenir directement à un noeud précédemment consulté et parfois d'introduire des 'stylets' (marquage de certains documents). Certains systèmes différencient nettement un mode création (auteur) d'un mode lecteur ('browser'), tandis que d'autres mélangent ces deux aspects. L'une des fonctionnalités centrale de ces derniers systèmes est l'annotation qui transforme un texte en un travail de collaboration réparti entre différents auteurs/lecteurs.

III.1.a.1.c Domaines d'application

La flexibilité des hypertextes leur permet de jouer un rôle important dans de nombreux domaines d'application classiques de l'informatique :

- Conception de livres électroniques (WEBS [PASQUIER & COLLAUD 87], EBOOK3 [SAVOY 87]), encyclopédies (Dynamic Medical Handbook à l'université de Washington);
- Documentation technique : nécessité de remplacer les documentations papier volumineuses (ex. 70 kg pour l'Airbus); on crée facilement des effets de zoom facilitant le passage du local au global, et on peut faire coexister plusieurs modèle;

- Maintenance : les parcours privilégiés ('guided tours'), en liaison avec les systèmes experts;
- Systèmes en ligne : aide, manuel de référence, voire tutoriel;
- Présentation : bornes interactives, communication;
- génie logiciel : commentaires de programmes, mise en commun de parties de code, etc.(travail en groupe, intégration, documentation, gestion des versions, systèmes DIF et NEPTUNE)
- Création collective de documents, annotation, etc.
- Littérature : à la fois du côté de l'explication, un texte littéraire étant par nature un hypertexte (c'est ce qui en fait la spécificité par rapport aux autres types de texte) et pour la création (textes construits par couches successives ou en collaboration) (voir III.2.b.3, page 95).
- Formation (voir III.1.b, page 85).

III.1.a.2 Exemples de systèmes hypertextes

III.1.a.2.a Historique

On fait souvent remonter l'histoire de l'hypertexte à un article de Vannevar Bush qui était conseiller du président Roosevelt et a proposé le système 'MEMEX', dont le but était de permettre aux scientifiques de pouvoir consulter de grandes quantités d'information de manière associative, mais ce système n'a jamais été implanté [BUSH 45]. Le nom même d'hypertexte est dû à Ted Nelson au début des années 60 dont le travail se poursuit autour du gigantesque système Xanadu [NELSON 81]. Engelbart est considéré comme un des pionniers en implantant les idées hypertextes [ENGELBART & ENGLISH 68] avec NLS. Il faut citer aussi les travaux de Kay et Goldberg sur le multi-media et le 'DYNABOOK' [KAY & GOLDBERG 77].

Pour donner une idée de la diversité des implantations hypertextes, voici brièvement trois systèmes importants, conçus suivant des points de vue très différents.

III.1.a.2.b NOTECARDS

Le logiciel NOTECARDS, créé à Xerox ([HALASZ 88] [IRISH & TRIGG 89]) se définit comme *« un environnement hypertexte extensible conçu pour aider les gens à formuler, structurer, comparer et gérer l'information »*.

C'est une généralisation électronique des bouts de papier sur lesquels on consigne des notes, qui permet l'analyse d'informations en les structurant et la production de rapports. NoteCards cherche à favoriser le travail en collaboration et les concepteurs soulignent que l'environnement doit permettre la méta-discussion, i. e. la discussion entre les collaborateurs sur le travail et sur l'utilisation du medium.

Malheureusement, le prix élevé du système (écrit en LISP sur des stations de travail) et sa complexité limitent ses potentialités de distribution en dehors de la communauté des chercheurs.

III.1.a.2.c HYPERTIES

Hyperties (où TIES est l'acronyme de *"The Interactive Encyclopedia System"*) [SHNEIDERMAN 89] [PLAISANT-SCHWENN 89] est construit sur un modèle d'encyclopédie : *« l'information est présentée en articles, chaque article couvrant un sujet particulier. Chaque article a un titre, un court résumé et un contenu pouvant couvrir plusieurs pages »*. Hyperties a été conçu à l'université du Maryland, où on étudie activement son utilisation (voir par exemple [MARCHIONINI 90]).

III.1.a.2.d INTERMEDIA

INTERMEDIA conçu à l'université de Brown se destine à l'enseignement dans un milieu universitaire. Voir [YANKELOVITCH 85], [YANKELOVITCH & Al. 85], [MEYROWITZ 89], [LANDOW 89a]. Deux cours principaux ont été développés : un cours de biologie et un cours de littérature. G. Landow [LANDOW 89a] décrit 'Context 32', complément à un cours sur la poésie victorienne, qui donne accès à un étudiant à l'environnement artistique, religieux et philosophique d'un auteur.

Parmi les systèmes importants, on peut citer KMS [AKSIN & Al. 88] développé à l'université de Carnegie-Mellon, conçu pour la collaboration à travers un réseau multi-utilisateurs. GUIDE de la société OWL le premier logiciel d'hypertexte diffusé sur micro-ordinateur. Enfin HYPERCARD de Bill Atkinson qui a popularisé la notion, à travers un produit essentiel à la stratégie commerciale de la société Apple.

III.1.a.3 Problèmes généraux

La figure ci-dessous présente un tableau général à partir de 12 caractéristiques qui permettent de situer les différents systèmes [ROMISZOWSKI 90].

System/Strategy Specification Profile			
Purpose	Access (reference) "Tool"	Gain (Learning) "Tutor"	Create Knowledge (Research/Author "Tutee")
Task/Content	Linking up existing documents	Adapt/Annotate existing documents	Originate the System
User Characteristics	Specific Objectives/Groups	Definable range of Obejctives/groups/user"	"Universal
Mode of use	Read-Only (Closed)	Read and Comment	Write/Read Write(Open)
Media Needs	Text only	Text + Graphics (+ sound)	Full Hypermedia
Node "Size" Needs	"Card(s)"	Page(s)	Article/Chapter
Links between Nodes	"Whole" Nodes Only	Words/Concepts of a Node	Concepts - "Typed" Links
Network Representation	Implied	Static Visual	Dynamic Model
Navigation (freedom of)	Constrained	Advice Help	Free
Navigation (freedom of)	Index	Online Help	"Intelligent" Help
Authoring Needs	Enhanced Word Processor + Database	HyperCardlike Environment	Multi-Tool Workstations
Evaluation Approach	User satisfaction	Specific Criteria Reference Learning	Cognitive Style/ Structure Change
	←		→

[ROMISZOWSKI 90]

III.1.a.3.a Modes de recherche

Au niveau lecteur ('browser'), l'hypertexte est avant tout un outil de recherche d'information. Les techniques utilisées sont liées aux modes d'organisation des connaissances et d'interface choisies [CARLSON 89] :

- Base de données :
  - choix par mots clés,
  - usage de menus encastrés.
- Science cognitive :
  - traitement de données spatiales (icônes),

  - filtres de 'vue distordue' : le programme génère un voisinage en calculant une relation sur l'importance a priori d'un élément dans la structure et la distance entre cet élément et la position courante.
  - browsers graphiques, i. e. cartes globales présentant la structure.
- Techniques de l'Intelligence Artificielle, on distingue 3 approches : (où l'hypergraphe désigne le réseau de liens)
  - séparer la base de connaissances de l'hypergraphe,
  - fusionner la base de connaissance dans l'hypergraphe,
  - encastrer ou distribuer les systèmes experts dans l'hypergraphe, c'est une sorte de combinaison des deux approches précédentes (avec une architecture multi-experts).

Il faut bien différencier gestionnaire de bases de données et hypertexte dans la structure associative de l'information et le contrôle dynamique offert à l'utilisateur [JONASSEN & GRABINGER 90].

La recherche et la navigation s'appuient sur deux composantes : une base de documents et un mécanisme pour retrouver les documents. Ces deux composantes sont reliées, le travail sur chacune d'elles est inversement proportionnel à celui effectué sur l'autre. Les hypertextes fortement structurés induisent des mécanismes de recherche très aisément, tandis que la recherche d'information dans des bases moins organisées est plus difficile.

On essaye ainsi d'intégrer de nouvelles techniques pour accroître les possibilités et ne pas s'enfermer dans un modèle trop restrictif. Voir par exemple [PEACHAM & WOOLLIAMS 90], le projet EMSBASE [MIDORO & OLIMPO 90] introduisant des techniques issues de l'IA et des bases de données, ou le travail d'André Le Meur qui défend l'idée de banque de données navigationnelle plus qu'hypertexte (SOKRATES banque en philosophie [LE MEUR 90]) :

« Montrer ... les fonctions nécessaires et les modes d'utilisation possibles de ce nouveau media qui, s'il reste à inventer, ne peut que s'inscrire dans la continuité des supports classiques (le livre et la banque de données) et de leur mode de représentation et d'organisation de connaissances. »
Les problèmes de communication et de portage invitent aussi à respecter les nouvelles normes et les langages de description de documents (SGML, ODA).

III.1.a.3.b Représentation des liens

La façon de représenter les liens à l'écran pour les rendre visibles à l'utilisateur est essentielle. Au niveau du texte [PLAISANT-SCHWENN 89], ils peuvent être inclus dans le texte lui-même et visualisés :

- soit par un attribut de caractère particulier (soulignement, surbrillance, etc.),
- soit par des icônes ou des symboles ajoutés au texte.

La première technique semble préférable :

- transparence du lien (le mot ou groupe de mots 'support' donne une idée plus claire de la destination de ce lien),
- absence de marqueurs, donc aucun intrus ne cache ou gêne la continuité du texte,
- simplicité de création qui s'effectue par marquage de mots ou groupe de mots,

- indépendance du lien par rapport à la disposition spatiale du texte : la modification du texte (ajout, insertion, suppression) ne modifie pas les liens qui sont directement accrochés.

Ceci donne aussi une structure supplémentaire sur le texte affiché en visualisant l'ensemble des liens et leurs supports, une première forme de réification (voir la notion de point de vue, III.2.b.5, page 98).

Au niveau du graphique, les techniques sont multiples mais n'ont pas la simplicité et la canonicité de celles utilisées sur le texte. On s'appuie soit :

- sur des icônes, qui sont ajoutées à l'image, et qui peuvent être cachées ou visibles,
- sur des zones d'écran marquées,
- sur des objets graphiques supportant les liens.

C'est cette dernière technique qui correspond le mieux aux liens textes, mais elles supposent des modes de description et de reconnaissance d'images encore peu développées qui impliquent de pouvoir séparer des constituants signifiants. Elle ne peut être utilisée avec les images classiquement manipulées en 'bit map'.

Quelle que soit la technique utilisée, il n'est pas souhaitable que les liens soient constamment visibles à l'écran (surcharge ou déformation de l'image). D'un autre côté, l'utilisateur peut vouloir savoir à quel endroit il peut obtenir des informations, surtout quand il commence à être perdu et qu'il essaye désespérément de cliquer sur des zones non actives. On ajoute ainsi des possibilités de visualiser les divers liens présents sur une image. Cette visualisation peut être assurée au passage de la souris (mode révélation), par une touche ou une sélection, ou automatiquement après une mauvaise sélection (c'est le cas d'Hyperties). Cette visualisation correspond à une inversion vidéo de la zone, un entourage ou à l'ajout d'une légende (ou une combinaison des techniques précédentes). HyperImages (SOFTIA 89) utilise des zones marquées auxquelles on associe deux modes de visualisation décidés par l'auteur.

III.1.a.3.c Lien avec l'I. A.

Il est important de noter [CARLSON 89] que l'hypertexte n'est pas une forme d'intelligence artificielle. L'IA essaye d'encapsuler la connaissance humaine dans un paradigme qui puisse être utilisé par une machine. L'hypertexte est destiné à augmenter la pensée humaine en fournissant une plate-forme dynamique pour gérer et présenter l'information.

Dans cette vision l'aspect pratique de l'hypertexte est ainsi mis en exergue. L'intelligence n'est pas dans la machine qui est essentiellement un outil, mais chez celui qui s'en sert. Il s'agit plus d'humaniser la technologie informatique que d'automatiser les processus cognitifs humains et les activités associées telles que l'écriture [BARRETT & PARADIS 88].

Un des problèmes classiques dans la conception des systèmes experts est celui de la récupération de l'expertise (extraction et codage pour l'utilisation par des professionnels) et celui de la diffusion de cette expertise au sein de l'entreprise, pour des gens d'une qualification moins grande. Il n'y a pas de modèle de référence (cadre conceptuel) vraiment partageable et on doit passer par une étape nécessaire de formalisation pour obtenir une représentation symbolique. L'hypertexte ne nécessite pas obligatoirement de recourir à une telle représentation, facilite grandement l'accès à la connaissance. Il peut aussi être un outil pour l'expert en favorisant un codage presque analogique des savoir-faire.

III.1.a.3.d Création d'un hypertexte

Ben Shneiderman [SHNEIDERMAN 89] énonce ainsi les règles d'or de l'hypertexte :

- il y a un large corpus d'informations organisé en nombreux fragments,
- les fragments sont reliés les uns aux autres,
- l'utilisateur a besoin d'une petite fraction à chaque fois. (Faire un lien avec ARRIA, même nature non linéaire) (voir aussi [LANDOW 89c])

Au delà de ces règles de bon sens, aucune méthodologie générale de conception des hypertextes ne peut encore être proposée. Le premier problème est de savoir comment construire le modèle de l'information : comment commencer, quelles informations choisir, comment les séparer, comme les séquencer?

Plusieurs modes de structuration peuvent être choisis [JONASSEN & GRABINGER 90] :

- structure sémantique, à partir de la connaissance de l'auteur ou de l'expert,
- structure conceptuelle, à partir de relations prédéterminées comme des taxonomies,
- structure reliée aux tâches, l'objectif étant de faciliter l'accomplissement d'une tâche,
- structure reliée à la connaissance, celle de l'expert ou celle de l'apprenant,
- structure reliée aux problèmes, i. e. on simule des problèmes ou des prises de décision.

On arrive ainsi à des hypermedia de type :

- non structuré (ce qui complique les processus de recherche),
- structuré avec une organisation explicite du type :
  - problème/solution,
  - chronologie, séquence,
  - partie/tout,
  - cause/effet, antécédent/conséquent, ...
- hiérarchique (i. e. hyper-structuré!).

L'objectif étant de faciliter la maîtrise d'un domaine à un utilisateur «naïf», il faut pouvoir penser ce domaine de manière non linéaire : on ne construit pas des progressions linéaires (elles peuvent exister comme des «coupures» ou des chemins privilégiés dans l'hypertexte). Il faut pouvoir analyser :

- sur le domaine lui-même :
  - l'ensemble des concepts,
  - prévoir des liens multiples :
    - * hiérarchiques : sous ou sur concept, englobant,
    - * proximité : proche ou contraire
  - profondeur sur le domaine
- en ce qui concerne l'utilisateur :
  - différents contextes d'utilisation : que veut-il savoir, quand, comment? A chaque problème connu on doit pouvoir associer une information, mais comment reconnaître dynamiquement ce contexte?

- ne pas présupposer des connaissances qui permettent de comprendre la hiérarchie des concepts du domaine,
- modes d'accès : minimiser le chemin d'accès à une information :
  * direct (lié à un mot ou un bouton)
  * hiérarchique (suite de menus).

Des outils destinés aux auteurs d'hypertexte commencent ainsi à apparaître, comme l'utilisation d'un réseau sémantique et divers algorithmes spécialisés facilitant la structuration et l'organisation de l'hypertexte. On peut créer des sortes de cartes sémantiques [HAMLIN & STEMP 90] en prenant un concept, en listant chacun des concepts auquel il est lié et en notant les forces et intensité de leurs relations.

Il faut aussi franchir deux écueils importants :

- la préparation des données : comment transformer du texte en hypertexte?
- la distinction auteur/lecteur : l'utilisateur doit-il pouvoir créer ses propres liens?

Neil Larson, auteur de PC-Hypertext, décrit rapidement la méthodologie qu'il suit pour créer un hypertexte à partir d'un document texte standard de 150 pages :

1. Récupérer les ressources à l'aide d'un système de lecture optique.
2. Nettoyer le texte obtenu : corrections orthograpiques et grammaticales, erreurs de lecture.
3. Séparer le texte en différents fichiers, ajouter des descripteurs pour chaque idée présente dans le texte. En gros, les 150 pages devraient se transformer en 500 fichiers.
4. Construire le fichier maître des références croisées.
5. Mettre les références croisées dans chacun des fichiers.
6. Vérifier l'intégrité du réseau créé : pas de fichier isolé, pas de liens sans destination, etc.
7. Création d'un arbre de décision pour construire les hiérarchies et réseaux de connaissances conduisant aux fichiers.

Neil Larson a d'ailleurs développé des outils spécifiques adaptés aux différentes étapes précédentes.

III.1.b Hypertexte et EIAO

III.1.b.1 Intérêt de l'hypertexte en éducation

P. A. Carlson [CARLSON 89] opère une distinction entre l'hypertexte idéal et l'hypertexte appliqué, qui oscille entre la liberté totale et la structure rigoureuse (voir une telle distinction en EIAO). Elle rapporte un exemple imaginé assez frappant.

Dans le premier cas, on suppose que l'enseignant donne à des élèves un très grand nombre de cartes d'index contenant le contenu du cours et demande aux élèves de parcourir les informations dans l'ordre qui leur convient. Les chemins suivis étant tous différents, il devient difficile de faire une évaluation commune. D'un autre côté, si l'enseignant donne des instructions rigoureuses concernant les cartes à lire et celles à contourner, les élèves verront

les mêmes informations. Cela ne s'accommode pas des styles individuels d'apprentissage. Il est clair que l'utilisation scolaire va se situer entre ces deux extrêmes.

Trois processus d'apprentissage semblent être le mieux supportés par les hypermedia [JONASSEN & GRABINGER 90] :

- la recherche d'informations,
- l'acquisition de connaissances (perspective de schémas),
- la résolution de problèmes.

La recherche d'informations comme méthode pédagogique a longtemps été négligée. L'information est vue dans le cours, les exercices et les problèmes font appel à cette information que l'élève a du apprendre, retenir et doit savoir mettre en oeuvre. (Dans LYRE, III.2.b.3, c'est une lecture méthodique qui doit être source d'informations). *« Il faut donc valoriser les stratégies d'apprentissage qui s'appuient sur l'envie d'apprendre ('need to know'), sur l'activité propre du sujet qui, pour réduire un manque (une ignorance) est mis en situation d'analyser son attente, d'identifier les ressources disponibles et d'intervenir sur cet environnement... Il (l'apprenant) lui faut s'engager dans une démarche d'évaluation du manque (ce qu'il veut savoir), d'identification des ressources potentielles et de leur organisation, puis dans une stratégie d'acquisition. »* [LE MEUR 90]

On distingue deux temps : localiser l'information (problème d'accès), puis, l'information étant trouvée, recherche de compléments ou d'éclaircissements. Ce processus étant récursif. Cette problématique de recherche d'informations est celle des systèmes d'aide (II.1.c.3, page 41).

" Learning vis-à-vis hypermedia is a process of accretion, restructuring and tuning. Hypermedia is the ultimate accretion medium, a knowledge base of interrelated ideas that can be readability accessed and assimilated. Restructuring is permittted by the dynamic control capabilities of creating and reorganizing links in the system. Tuning is provided by the authoring and collaborative characteristics of hypermedia. " [JONASSEN & GRABINGER 90]

- accrétion : accumulation d'informations pour remplir les schémas,
- restructurer : ajouter des schémas ou de nouvelles conceptualisations,
- 'tuning' : adaptation à des tâches précises.

Enfin, la résolution de problèmes peut être facilitée par l'utilisation des hypermedia à différents niveaux :

- représentation du problème,
- transfert,
- évaluation.

Si on considère que dans un hypertexte, l'utilisateur doit pouvoir créer ses propres liens, i. e. si on ne le limite pas à un outil de traitement de l'information, mais comme un environnement pour le développement des connaissances et l'exploration de problèmes ouverts, les deux points centraux sont :

- la nature collaborative du travail,
- le processus de réification sur la production d'écrits.

Comme le souligne Barrett [BARRETT 89], l'important n'est pas de théoriser sur notre environnement mental personnel (sauts associatifs, inspiration divine ou opérations booléennes), car l'environnement virtuel de l'hypertexte nous permet d'objectiver nos idées dans l'écriture de nouveaux textes.

L'aspect collaboration est cohérent avec les bénéfices que les enseignants semblent accorder à l'utilisation des machines (II.3.a.5, page 71), et cette réification introduit une nouvelle forme de micromonde (II.1.b.2.a, page 33).

E. Barrett souligne d'ailleurs la filiation avec Papert [BARRETT 88] et met en garde contre des visions anthropomorphiques de l'ordinateur : ce n'est pas un objet pensant, mais un moyen d'objectiver et d'afficher le processus cognitif d'un apprenant. L'utilisateur n'est pas programmé par la machine mais la programme et acquiert des connaissances au travers de son activité.

III.1.b.2 Exemples

Contrairement à la plupart des tuteurs intelligents (II.3.a, page 67), les systèmes hypertextes sont fortement utilisés en formation, surtout dans un contexte universitaire. Ils induisent une réflexion sur les pratiques et les buts de l'Education.

G. Landow rapporte le constat fait par Putka : " *Independent of hypertext, dissatisfaction with american secondary school student's ability to think critically has recently led to a new willingness to try evaluative methods that emphasize conceptual skills - chiefly making connections - rather than those that stress simple data acquisition.* "

A partir de l'utilisation d'Intermedia, et plus particulièrement de 'Context 32', Landow précise les améliorations apportées par l'hypertexte [LANDOW 90] :

- développe la pensée critique,
- habitue les étudiants à faire leurs propres connections,
- encourage le travail en collaboration,
- et, dans le cadre de grandes bases, favorise un travail interdisciplinaire. Ceci amène un impératif qui est de faire le cours en intégrant l'utilisation de cet outil (voir II.3.a.5, page 71).

Les expériences font aussi état de divers problèmes montrant que l'utilisation de ce nouveau type d'outil ne va pas sans problèmes (voir III.3, page 103). Pour le système EOS (Educational On-line System) [BARRETT & PARADIS 88], les élèves regrettent une interface utilisateur trop complexe et rejettent, pour la plupart, l'annotation en ligne , pour des raisons de mélange avec le texte initial. Des expérimentations conduites avec un hypertexte de présentation (HyperTIES) pour un cours sur l'assembleur et un sur le langage Modula-2 [LEGGETT & Al. 90] amènent les constatations suivantes :

- dans les deux cas, les étudiants ont trouvé les livres plus faciles à lire et plus accessibles que les tutoriels hypertextes.
- ils désirent des possibilités d'annotation et d'introduction de 'signets'
- ils voudraient pouvoir disposer du tutoriel dans l'environnement de programmation (Ceci est partiellement fait avec APILOG, III.2.b.2, page 93, et complètement avec HyperInfo, III.2.c, page 99).

Les chercheurs (voir par exemple [MARCHIONINI 90]) commencent à réfléchir sur les modes possibles d'évaluation de l'apprentissage à l'aide de systèmes hypertextes. On retrouvent des difficultés comparables à celles que l'on rencontre avec les tuteurs intelligents (II.3.a, page 67).

III.1.b.3 Langages auteur et hypermedia

Dans le cadre de la formation professionnelle, l'utilisation de techniques multi-media rencontre un plus grand succès que l'intégration des techniques issues de l'intelligence artificielle. On peut y trouver diverses justifications :

- le multi-media s'intègre bien dans la tradition des langages auteur traditionnels (voir IV.3.c.1, page 140),
- la formation dispensée est souvent plus proche de l'information que de la formation proprement dite,
- un aspect extérieur attrayant est reconnu comme un facteur important de motivation,
- les développements minimisent les connaissances informatiques nécessaires,
- l'apprentissage par l'action est encore rare.

Ainsi, Geri Younggren [YOUNGGREN 88] décrit HyperCard comme le résultat d'une convergence entre les hypermedia et les environnements de programmation visuelle, prototype des langages auteurs destinés aux enseignants. Il ne faut cependant pas oublier le fait que la simplicité initiale d'un outil comme HyperCard contribue largement à son succès, mais que des développements importants nécessitent la maîtrise de concepts généraux et informatiques, trop souvent ignorés des 'informaticiens occasionnels' : faire simplement des choses simples ne permet pas automatiquement de transférer ces savoir-faire locaux à des réalisations plus ambitieuses. Cette distinction est d'ailleurs généralement bien intégrée par les professionnels et l'attrait des outils hypermedia comme systèmes auteur s'associe à l'idée que les formateurs veulent créer des logiciels ciblés et peu complexes, bien adaptés pour compléter leur enseignement, et réclament des outils de création simples et performants.

Les hypermedia et les systèmes auteur se rejoignent dans la problématique de gestion d'un curriculum. Les techniques utilisées sont voisines, découpage en unités et recherche des liens entre ces unités (IV.3.c.1, page 140). Ainsi, un hypertexte construit à partir de liens entre des concepts se rapproche du modèle 'BiteSize' de Lesgold [LESGOLD 85].

Les langages auteur récents incluent des ressources hypertextes, ce qui autorise des intégrations multiples :

- un système ouvert d'accès aux connaissances,
- des tuteurs locaux (avec une gestion globale hypertexte),
- des ressources en ligne additionnelles sur des environnements traditionnels,
- etc.

Ces nouvelles techniques facilitent et induisent de nouvelles formes de production par collaboration : on arrive au concept d'autorat distribué. Des environnements de création incluent des ressources multiples reliées pour aider l'auteur dans son travail (voir par exemple IDE ou ALEXANDRIA [RUSSELL 90]).

Dans le cadre d'une pédagogie active, se développe le concept de Livre/Cahier où l'annotation et la création de liens à partir de documents déjà structurés constituent une des activités moteur de l'apprentissage (voir micromonde/ hypertexte).

Chapitre III.2

Exemples d'utilisation en formation

III.2.a Divers types d'utilisation de l'hypertexte

Ce chapitre montre d'abord quelques réalisations de didacticiels utilisant les techniques hypertextes. Ces dernières interviennent comme ressource supplémentaire aux outils classiques de l'EAO ou de l'EIAO, dans le cadre d'un apprentissage basé sur l'activité ou la résolution de problèmes [BRUILLARD & WEIDENFELD 90] :

- représentation de l'ensemble des connaissances du domaine, sous une forme cependant non exécutable, en quelque sorte la partie cours;
- explicitation des messages du système :
  - questions et consignes dont la formulation peut être ardue, i. e. aide en ligne,
  - messages d'erreur ou précisions complémentaires;
- activité de lecture elle-même (voir LYRE, III.2.b.3, page 95).

En ce qui concerne l'explicitation des messages d'erreur, on retrouve ici une différence essentielle entre l'approche hypertexte et l'Intelligence Artificielle. Un système de diagnostic ne pouvant être ni complet ni réellement fiable, on cherche à indiquer la source la plus probable de l'erreur en donnant accès à l'utilisateur à **toutes** les précisions qui semblent utiles dans le cadre où il se trouve. On suppose qu'en général, l'utilisateur sait mieux que le système ce qu'il ne sait pas et on lui donne une chance de rétablir un bon fonctionnement (voir modèle élève, V.1.b, page 152). En fait, on établit plus une forme de collaboration qu'une forme tutorielle.

Une deuxième partie traite le cas plus spécifique des systèmes d'aide en ligne à travers HyperInfo, où l'hypertexte est une ressource supplémentaire dans son implantation même, couche ajoutée à une autre application.

Enfin, l'aspect création montre l'hypertexte vu comme un réexamen des connaissances, i. e. le passage d'un mode d'exposition séquentiel à un modèle relationnel.

III.2.b Exemples de réalisation

III.2.b.1 Le système SEVE

Initialement, SEVE a été conçu au Centre Mondial de l'Informatique dans le cadre du groupe didacticiels dirigé par Gérard Weidenfeld. Il s'agissait de construire un système général d'explications de documents administratifs et de fournir une aide efficace aux divers types d'usagers concernés. A la fermeture du Centre, le système a été achevé et porté sur compatible PC dans le cadre de la société Softia. L'intuition à la base de cette réalisation est la reconnaissance de l'importance primordiale de disposer d'un environnement adaptatif pour expliquer des textes à diverses classes d'utilisateurs. Ceci s'applique aussi bien pour le remplissage de divers documents que dans des activités de résolution de problèmes où beaucoup d'élèves sont bloqués à cause d'une incompréhension de l'énoncé.

SEVE est un système écrit en langage C présentant les caractéristiques suivantes ([BROUAYE & Al. 87a] ou [WEIDENFELD 87]) :

- un document structuré hiérarchiquement avec des pages, des chapitres et des sections,
- des champs de saisie pouvant correspondre à plusieurs lignes avec un éditeur associé,
- des explications avec un hypertexte et un système de fenêtres, ces explications pouvant être associées à des mots ou expressions mises alors en surbrillance et aux divers champs;
- des points de vue associés à l'hypertexte, permettant de changer les explications et de transformer dynamiquement le document (surbrillances différentes);
- un langage Prolog écrit également en C et communiquant complètement avec le reste du système pour les entrées/sorties : des programmes pouvant être attachés à des touches, des champs ou des entités;
- un gestionnaire de menus et un petit gestionnaire de bases de données.

Il ne s'agit donc en aucun cas d'un langage auteur au sens classique du terme mais plutôt d'un système de développement riche et ouvert intégrant un langage de programmation complet et puissant (Prolog) et un générateur d'hypertextes très simple d'emploi.

L'utilisation de ce système pour la réalisation de logiciels d'enseignement conduit à diverses constations [BRUILLARD 88] :

- SEVE est un système complexe, pas au niveau du développement proprement dit, mais au niveau de la conception. Pour l'utiliser pleinement, il faut écrire des hypertextes (penser à des accès non linéaires à l'information) et écrire des programmes Prolog.
- On dispose d'une description abstraite des fonctionnalités du système, correspondant à des types d'interaction encore peu classiques, il faut les traduire et leur donner du sens dans un cadre disciplinaire.
- L'utilisation de Prolog est réellement significative si on arrive à donner de la connaissance au système, malheureusement dès qu'on s'attache à un problème particulier, la connaissance nécessaire est rarement formalisée.
- Les enseignants non informaticiens ont souvent soit une représentation un peu datée des capacités des ordinateurs, soit une vision trop naïve et futuriste. Ils ont ainsi tendance à essayer de reproduire sur machine leur enseignement habituel et à imaginer

des interactions trop pauvres (style question-réponse) ou trop loin des possibilités techniques actuelles (compréhension de toutes les subtilités du langage naturel) (voir modes d'usage).

Ainsi, il faut trouver des auteurs experts dans leur domaine, capables d'apporter des solutions à des problèmes réels d'apprentissage, et tenter de trouver avec eux une solution dans un travail d'équipe. N'ayant pas de structure entièrement prédéterminée à leur proposer (comme dans un langage auteur), il est nécessaire d'effectuer un transfert de leurs connaissances via un nouveau médium, ce qui entraîne immanquablement un réexamen de ces connaissances. Ce mode de travail impose une forme de synergie entre SEVE et les domaines traités, ceux-ci devant donner un sens intrinsèque aux fonctionnalités abstraites du système. Il faut en effet se débarrasser de l'illusion de l'universalité d'un système dès qu'on essaye d'intégrer les spécificités d'un thème d'application (voir langages auteurs, IV.3.c.1, page 140). Les exemples qui suivent montrent quelques réalisations effectuées avec le systèmes SEVE (APILOG, LYRE, AIP, ARRIA).

III.2.b.2 APILOG

III.2.b.2.a Description d'APILOG

APILOG est un programme d'auto-formation au langage Prolog conçu par une équipe interne de Softia. Il se divise en 3 parties : une partie cours comprenant 5 chapitres, un manuel sur les prédéfinis et 7 exemples de programmes commentés.

Disposer d'un Prolog pour enseigner le langage Prolog est ici le trait principal. L'utilisateur apprend Prolog en en faisant, ce qui règle la plupart des problèmes de plus bas niveau et évite une séparation trop grande entre la phase d'appropriation et la mise en application éventuelle.

Les fonctionnalités SEVE sont ici utilisées d'une manière tout à fait classique (ce qui n'est pas étonnant vu que le domaine est purement informatique) :

- l'hypertexte gère l'interface active de lecture et donne une documentation en ligne de tous les prédéfinis et des concepts généraux de Prolog, accessibles directement pour la conception des programmes. Les points de vue permettent de différencier deux profils distincts (utilisateur débutant en informatique ou connaissant déjà un peu de programmation). D'une manière plus subtile, ils permettent de donner plusieurs éclairages différents à un même programme Prolog : en particulier, suivant les instanciations initiales, le comportement d'un programme incluant des générations et des tests peut radicalement changer, un même code étant utilisé dans des contextes distincts (voir C.a, page 223).
- l'éditeur sert à entrer les paquets de clauses et les requêtes.
- Prolog sert à la fois de langage cible et de vérificateur de certains exercices.

Par rapport à un tuteur comme le LISP TUTOR [ANDERSON & REISER 85], nous n'avons pas introduit d'analyse fine des erreurs de programmation, en restant au niveau de la syntaxe. En effet, APILOG avait débuté au Centre Mondial comme projet de type universitaire, une commercialisation rapide a dû en restreindre les objectifs.

III.2.b.2.b Remarques sur l'utilisation d'APILOG

APILOG a été étudié par deux trinômes à l'Ecole Supérieure d'Electricité durant l'année scolaire 1987-1988. Le but de leur travail consistait à évaluer le logiciel, non à apprendre le langage Prolog, et à le situer par rapport à d'autres modes d'apprentissage.

Leur bilan est plutôt négatif et s'appuie sur plusieurs remarques :

- l'ergonomie est insuffisante, le programme est parfois lent, le nombre de touches à utiliser est trop important, l'interface est un peu complexe (ceci est dû à des insuffisances du programme ainsi qu'à des contraintes matérielles inévitables : écran trop petit, absence de la souris, fenêtres trop petites, etc.).
- difficulté de circulation entre les chapitres (pas à l'intérieur d'un même chapitre).
- préférence donnée au livre (à relier à III.1.b.2, page 87).

Un groupe suggère même dans les améliorations du logiciel une fonctionnalité déjà implantée mais qu'il n' a pas su utiliser, ce qui montre une incapacité du système à faire comprendre ses possibilités qui induit de fortes limitations d'usage.

Une de leur réflexion est très indicative de leur état d'esprit : « *Le cours magistral est inégalable pour son efficacité pédagogique. Le maître non seulement communique un contenu riche (paroles mais aussi gestes et dessins explicatifs) mais encore il stimule l'assemblée par son authenticité vivante, ses gestes focalisateurs, sa tonalité fluctuante,... En bref, il retient l'attention, de plus il est la preuve criarde (sic) que ce savoir est accessible à l'entendement humain.* »

La démarche analytique 'page par page' qu'ils ont utilisée est un peu antinomique de la conception même du logiciel. Ils aiment en général les exemples et les exercices, mais ne gèrent pas leur parcours, reviennent rarement en arrière, n'explorent jamais en avant (ceci est peut-être dû à leur idée de test). Un groupe fait une étude détaillée du programme en suivant systématiquement le déroulement séquentiel du programme (pour être sûr d'évaluer toutes les possibilités). Ceci illustre certaines difficultés méthodologiques : comment évaluer en suivant un parcours linéaire, un document conçu pour être utilisé de manière non linéaire.

Un groupe reconnaît qu'Apilog leur a permis d'avoir une vision synthétique de Prolog, et le situe, avec juste raison, comme un outil de sensibilisation ou d'initiation.

Une objection formulée est celle de l'incomplétude : il n'y a pas de dialogue réel avec la machine, ce qui peut amener des questions sans réponse ou soulever des ambiguïtés. La machine seule est insuffisante, il est indispensable de pouvoir, dans certaines circonstances, recourir à un expert. Les deux groupes ont ainsi rencontré des difficultés et leurs interrogations n'ont pu être satisfaites que par un dialogue avec un professeur 'humain'. Cette restriction est assez constante (voir IV.2.d, page 132), on raisonne trop souvent en terme de solutions exclusives, non complémentaires (homme ou machine, non homme et machine).

Ceci peut amener deux conclusions :

- ne pas faire sortir trop tôt des programmes novateurs, s'ils n'ont pas atteint une maturité suffisante (ce qui n'est pas le cas d'Apilog).
- assister l'utilisateur dans l'usage de ces nouveaux outils (III.3.a, page 103).

APILOG a aussi été évalué par l'Education Nationale, mais je n'ai pu récupérer aucun rapport. Je n'ai pu rencontrer qu'une personne qui s'en est inspiré pour ses propres cours de Prolog!

Pour se consoler, on peut citer une critique plutôt aimable de Rajben dans 'Décision Informatique' : « *Optimisable, certes, en vivacité d'exécution et en aisance de circulation, Apilog est une réalisation triplement exemplaire : ce logiciel ouvre une voie, montre la juste direction et crée un pôle de référence.* »

III.2.b.3 LYRE

III.2.b.3.a Description de LYRE

LYRE est système d'aide à l'analyse de textes littéraires pour la rédaction d'un commentaire composé. Il a été conçu avec Mme Marina-Médiavilla, à partir d'une méthode qu'elle a mise au point et qu'elle enseigne depuis de nombreuses années ([MARINA-MEDIAVILLA]). « *Le principe de base de la méthode consiste à reculer le plus longtemps possible la rédaction du commentaire, afin de prendre le temps d'étudier le texte. Un commentaire doit être 'composé'; 'composer', c'est organiser des éléments; ces éléments, ce sont des résultats obtenus en étudiant le texte, les remarques faites en analysant les procédés employés par l'écrivain.* »

Cette méthode se décompose en 3 moments :

- l'étude du texte, qui s'effectue un crayon à la main pour noter au fur et à mesure les résultats des recherches,
- l'élaboration du plan, pour lequel on peut découper et regrouper les diverses remarques faites durant la première étape,
- la rédaction du commentaire composé.

LYRE intervient surtout pour le premier moment de la méthode, mais fournit aussi un bloc-note pour consigner les remarques faites au cours des différentes lectures du texte (remarques indexées automatiquement par le contexte de lecture courant). Les notes peuvent directement être récupérées dans un traitement de texte quelconque, donnant ainsi l'ossature du commentaire composé. On peut d'ailleurs aussi travailler avec un traitement de texte appelé directement à partir de LYRE.

L'étude du texte se décompose en 5 objectifs de lecture :

- étude des réseaux lexicaux et sémantiques;
- étude des images;
- étude du niveau phonique et prosodique;
- étude du niveau syntaxique;
- le niveau sémantique.

On peut aussi croiser divers points de vue de lecture en les visualisant simultanément sur le texte initial et effectuer une recherche transversale à l'aide d'un point de vue global. Dans ce cadre, l'hypertexte associé à la notion de point de vue est l'outil qui colle le mieux au concept de lectures plurielles : le texte littéraire est par essence un hypertexte! On dispose d'une grande quantité de commentaires, directement accessibles à partir du texte initial toujours présent, celui-ci étant le vrai déclencheur (voir C.b, page 227).

Un dictionnaire général directement consultable contenant les termes «techniques» du commentaire littéraire précise et enrichit les explications sans les rendre trop ardues.

Ce mariage entre hypertexte et texte littéraire a été remarqué par d'autres auteurs, travaillant dans des axes similaires, par exemple J. M. Slatin, qui voit dans cette relation une difficulté irréductible de la notion d'hypertexte :
" It is in its embodiment of intertextuality that hypertext is most closely analogous to poetry; and one implication of this analogy, I think, is that there is likely to be an irreductible element of difficulty about hypertext, as there is in poetry - neither form being meant to be easy. It is in the nature of hypertext to be complex and difficult, and we ought to respect that and work with it, rather than treating it as a flaw to be overcome. " [SLATIN 88] (voir III.3.a.1, page 103).

On peut remarquer qu'un aspect important qui manque dans LYRE est celui de l'impossibilité de mettre en relation différents textes, i. e. de travailler sur la littérature comparée. L'environnement d'Intermedia (III.1.a.2.d, page 80), dans ses possibilités d'exploration de divers documents, est d'ailleurs très complémentaire (vision en largeur vs vision en profondeur).

III.2.b.3.b Remarques sur l'utilisation de LYRE

Les réactions autour de LYRE sont variées, mais montrent un écart encore important entre ce type de réalisation et les attentes des enseignants. LYRE est conçu comme un outil complémentaire à un cours traditionnel. La place du logiciel est dans un centre de documentation ou à la maison, mais pas dans la salle de classe. Il ne veut pas et ne peut pas se substituer à l'enseignant traditionnel, mais permettre un entraînement à l'analyse de textes par l'intermédiaire d'une méthode générale, séquençant les différentes étapes de la rédaction d'un commentaire composé. L'interface est ici plus conviviale qu'Apilog et la circulation pose peu de problèmes. Le seul point noir concerne l'impossibilité de revenir en arrière dans le parcours en pas à pas de la méthode. Les controverses sont principalement de nature culturelle :

- cela n'apporte rien de plus que le livre : en effet, la différence est plutôt subtile en première approche, puisqu'elle se situe au niveau des modes de visualisation et d'accès aux commentaires. Il faut y ajouter cependant les possibilités de croisement, qui, à partir de 10 points de vue, permettent d'afficher 100 vues différentes, ce qui semble difficilement réalisable dans un livre!
- c'est un outil trop limité : il ne prend sens que si l'on suit toute la chaîne, de l'analyse au commentaire en passant par le plan, i. e. si l'on accepte les copies construites avec un traitement de texte, ce qui est loin de faire l'unanimité chez les enseignants (sous le prétexte fallacieux de copiage!).
- il n'y a pas d'interactivité : cette notion est souvent confondue avec un aspect tutoriel ou prise en charge de l'utilisateur. LYRE effectue peu de sollicitations, la motivation est censée être externe au logiciel. On retrouve encore le fantasme du questions/réponses, alors que dans un tel contexte, toute analyse sérieuse des réponses fournies est totalement exclue et ne peut conduire qu'à une illusion de compréhension.

  Remarque : cette critique est souvent peu cohérente puisque les logiciels de questions/réponses sont violemment combattus pour leur interaction trop souvent caricaturale, et par les mêmes personnes qui reprochent ce 'manque d'interactivité'. Le jugement s'effectue en comparaison avec un enseignant idéal, sans prise en compte des spécificités du travail avec un ordinateur. Cela dénote malheureusement surtout une certaine absence de réflexion.

- le dernier point concerne la complétude et l'objectivité. Les explications et commentaires fournis dans LYRE sont ceux d'un enseignant. A ce titre, ils correspondent à sa propre analyse du texte qui peut être controversée. Cette subjectivité, admise pour les livres, est ici refusée aux logiciels (voir III.2.c.1, page 100). Les points de vue de lecture sont uniquement ceux qui ont été choisis et il n'y a pas possibilité d'en rajouter de nouveau. Ceci n'est ni permis pour les élèves (ce qui restreint leur liberté et peut être considéré comme inacceptable, voir la remarque de J. Nielsen, III.3.a.1), ni pour les enseignants, qui ont néanmoins tout loisir d'ajouter un commentaire global. En France, cette restriction pour les élèves semble plutôt une obligation (pour éviter la réification de points de vue très discutables), mais semble plus difficile à admettre d'un point de vue théorique pour les enseignants. Au niveau pratique, ceci paraît pourtant une objection peu fondée, les enseignants étant encore peu enclins à effectuer ce genre d'intervention, qui n'est d'ailleurs pas possible avec un livre (on garde la liberté du choix, comme pour le logiciel).

J'ai pu remarquer que les personnes auxquelles on a pu faire une présentation du logiciel, en montrant bien ses possibilités et ses limites ont en général un avis plutôt favorable, tandis que ceux qui l'ont vu par l'intermédiaire d'autres personnes sont peu convaincues. Une hypothèse vraisemblable est que ces dernières présentations introduisent LYRE dans un cadre EAO traditionnel qui conduit aux objections mentionnées ci-dessus.

III.2.b.4 Autres réalisations

Un didacticiel sur l'étude de textes philosophiques a aussi été réalisé d'une manière assez proche de celle de LYRE. Malheureusement la réalisation est plus discutable, la forme des textes n'ayant pas les caractéristiques des textes littéraires, les liaisons avec l'hypertexte ne sont pas de même nature, les articulations sont moins convaincantes. Une forme un peu bâtarde de questions/ réponses a du être introduite, l'absence de liens entre de nombreux textes philosophiques conduisent à un produit peu satisfaisant.

L'application AIP (Avis d'Incident de Paiement) développée dans un cadre de formation professionnelle amène un éclairage un peu différent. Cette application a été mise en place dans les centres de chèques postaux. Son objectif est de faciliter le transfert des compétences des agents s'occupant du traitement des chèques impayés d'une procédure manuelle à une procédure partiellement informatisée. Les contraintes des gros systèmes ne permettant pas directement d'assurer des applications conviviales, contrairement à de nombreuses réalisations disponibles sur micro-ordinateur, le portage de cette application sur micro-ordinateur vise à entraîner à moindres frais sur des cas typiques (simulation de l'application réelle), une couche hypertexte donnant accès à toutes les informations utiles aussi bien sur l'application elle-même que sur la procédure en cours, ses justifications et ses conséquences. Ainsi, ce programme a une double vocation :

- simulation de l'application réelle,
- assistance à la formation des agents.

Cette assistance se situant sur trois plans :

- apprentissage des manipulations au clavier,
- compréhension des informations affichées à l'écran,

- étude des cas proposés par référence à la réglementation applicable.

Le logiciel comprend deux parties : la simulation du programme réel décrite ci-dessus, et un éditeur spécial pour rentrer de nouveaux cas, destiné aux formateurs de la poste pour adapter le logiciel aux situations locales et l'améliorer progressivement.

Le programme AIP a été très bien reçu et a rempli les objectifs pour lesquels il était destiné. On peut y voir plusieurs raisons :

- le cadre d'utilisation a pu être défini de manière précise, et le logiciel a été intégré entièrement dans la formation en association avec d'autres techniques : formation frontale, diaporama, travail sur l'application réelle;
- les exigences sont ici plus pragmatiques : il s'agit de formation continuée et plus précisément d'un transfert d'expertise (d'une procédure manuelle à un traitement informatisé);
- le programme est plus conçu en fonction des utilisateurs pour les aider à surmonter des difficultés que pour satisfaire des décideurs.

III.2.b.5 La notion de point de vue

LYRE illustre l'une des spécificités de l'hypertexte associé à SEVE qui est la notion de point de vue. Ceci confère au texte affiché à l'écran un aspect dynamique en relation avec un contexte de lecture particulier : l'objet ne change pas mais il apparaît différemment. Ce contexte dépend pour l'essentiel de trois paramètres :

- le profil de l'utilisateur, i. e. ses connaissances générales sur le domaine considéré. Ainsi APILOG intègre un point de vue DEBUTANT et un point de vue dit STANDARD, les explications associées étant différentes;
- le type de lecture choisi, ce qui correspond aux modes d'analyse du texte dans LYRE;
- l'historique de la consultation, i. e. le chemin suivi et ce qui a été réalisé.

Les deux premières utilisations des points de vue sont plutôt statiques et correspondent à des choix qui sont faits le plus souvent par l'utilisateur, la dernière est dynamique et doit être gérée par le système.

Il ne faut pas limiter la notion de type de lecture aux seuls textes littéraires. Ceci est tout aussi valable pour un programme Prolog par exemple ou un énoncé de problème de mathématiques. Ainsi, un même programme Prolog peut avoir des sens très différents suivant les instanciations des variables. Dans un programme de mastermind (voir C.a, page 223), le même code est utilisé pour proposer une combinaison, calculer un score et chercher une combinaison compatible avec un essai et un score donné.

L'énoncé d'un 'Word Problem' (II.2.b.3, page 61) peut être lu de diverses façons suivant la question que l'on cherche à résoudre (voir C.c, page 229). La mise en évidence des éléments pertinents est essentiel, la représentation même d'un énoncé et la façon dont il est lu influent de manière décisive sur le mode de résolution du problème associé.

Cette notion de point de vue renforce la réification due à la visualisation des liens hypertextes (voir réification). Elle donne une forme de vie à de l'écrit considéré inerte et induit des lectures non univoques intégrant de multiples significations. Elle est très proche de la notion de point de vue qui se développe dans les ITS [MOYSE 89] : mise en évidence dynamique d'un objet ou

d'une représentation particulière d'un objet dans le cadre d'un but spécifique. On représente la même information sous différents formalismes d'où l'obtention de modes complémentaires d'analyse qui peuvent être requis dans un contexte donné, référant aux mêmes objets mais les identifiant et les structurant de façons différentes.

« Contrairement au discours verbal, la représentation visuelle n'est pas limitée par ce dont une personne est capable de se rappeler du début à la fin d'un discours. La juxtaposition de plusieurs concepts et de leurs relations est analogue à un système; le tout est plus que la somme des composantes prises séparément. » [MEILLEUR & MITCHELL 89]

III.2.c Hypertexte et aide en ligne

La difficulté de conception des applications avec SEVE [BRUILLARD 88] nous a amené à séparer les diverses composantes du système, et, en particulier, à isoler l'hypertexte. Une première réalisation nommée HyperGuide ne conservant que les documents et l'hypertexte incluant la notion de point de vue a été menée. La lourdeur rémanente de ce logiciel a conduit à une nouvelle implantation réalisée par Paulo Machado, appelée HyperInfo.

HyperInfo est un générateur qui permet de créer des hypertextes. Ceux-ci se présentent comme un ensemble de fiches que l'on peut visualiser dans des fenêtres. Chacune de ces fiches comporte des mots clés qui sont autant de liens vers d'autres fiches que l'on peut ouvrir. Une caractéristique essentielle est que l'on dispose d'un utilitaire de lecture résident, c'est-à-dire capable de travailler en même temps qu'un autre logiciel. L'idée est de favoriser une démarche naturelle : intégrer la recherche d'information au processus de travail. Cette information est généralement une aide liée à la tâche en cours, aide qui peut être :

- contextuelle, i. e. l'utilisateur n'est pas forcé de partir d'un sommaire et peut accéder directement à l'information qu'il recherche,
- adaptable à un objectif d'utilisation : on peut travailler avec plusieurs hypertextes et en choisir un suivant le but poursuivi,
- personnalisable, on peut aisément compléter un hypertexte fait par quelqu'un d'autre pour l'adapter à ses propres besoins en y intégrant ses proches fiches ou en ajoutant de nouveaux liens.

On peut séparer les applications en deux grandes classes :

- celles où l'aspect résident est essentiel, leur caractéristique est de permettre à un utilisateur de bénéficier d'une couche d'aide ou d'information complémentaire à la tâche qu'il est en train d'exécuter.
- celles où l'aspect résident n'est pas fondamental, mais où on utilise l'hypertexte comme intégrateur ou comme livre électronique.

Deux exemples ont été étudiés plus en détail :

- HyperInfo Logo (E. Bruillard), aide en ligne sur le langage Logo, (voir C.d, page 230),
- HyperInfo Dos (Claude Drouin), aide en ligne sur MS-DOS.

Dans chacun de ces deux cas, le mode d'accès aux fiches est primordial, puisqu'à chacune d'elles on associe un nom et éventuellement des synonymes et chaque primitive ou mot clé

du langage est relié à une fiche. Pour lire la fiche correspondante, il suffit de taper ou d'avoir sur l'écran le mot considéré, ou même plus simplement le début de ce mot (si on hésite sur l'orthographe complète). On dispose par ce moyen d'une aide réellement contextuelle avec accès quasi direct à la plupart des informations. Par exemple, si on veut copier un fichier sur disquette en utilisant MS-DOS :
On tape
`>copi` (puis `SHIFT-F2`)
Une première fiche apparaît, indiquant les divers modes de copie possibles l'utilisateur peut alors explorer divers sous-fiches pour trouver l'information utile pour résoudre son problème. Une remarque importante : les fenêtres sont déplaçables ce qui permet de voir à la fois ce que l'on a tapé et les indications fournies dans les fiches consultées. A la fin de la consultation, on revient sous DOS pour compléter et lancer la commande voulue. Il faut noter que cela concerne aussi bien les débutants peu habitués à la manipulation des diverses commandes MS-DOS que les experts qui n'ont pas toujours en tête certaines commandes ou syntaxes rares et qui peuvent trouver directement un point particulier.

Des tests effectués à l'Ecole Normale avec ces deux programmes montrent qu'il subsiste encore certains problèmes d'usage (voir III.3). Leur création a permis néanmoins de dégager une méthodologie relativement générale de conception d'aide en ligne pour des langages ou des systèmes de commande (II.1.c.3, page 41) :

- organisation autour des différentes commandes ou primitives, avec des modes d'accès hiérarchiques standards,
- introduction de synonymes sur chacune des fiches précédentes tenant compte des noms plus habituels que les utilisateurs risquent d'employer,
- même organisation sur l'ensemble des messages d'erreur en précisant les contextes où ils surviennent et en fournissant des explications sur les causes connues et les remédiations à assurer,
- introduction des modes d'accès par activité (sauvegarder, créer une procédure, une variable, etc.),
- introduction des modes d'accès par type de problème généralement rencontré. Cette organisation donne une grande souplesse d'accès aux informations utiles et facilite les références entre les fiches. Ceci illustre aussi un mode de création qui s'appuie en même temps sur la structure du domaine et les difficultés des apprenants.

De la même manière, on construit aisément des dictionnaires : constitution de glossaires spécialisés, déclaration de synonymes, thésaurus,... (voir un exemple de grammaire en ligne, C.e, page 234). Par exemple, l'auteur du traitement de texte pédagogique le «SCRIPTEUR», Christopher Hopper de l'université de Montréal, intègre des dictionnaires analogiques (idées proches) pour favoriser l'imagination dans la rédaction des textes. Deux dictionnaires ont été terminés : un sur le conte, l'autre sur l'alimentation.

III.2.c.1 Hypertexte : outil de création

L'utilisation de l'hypertexte comme outil de création récèle d'énormes possibilités dans l'éducation : création d'un savoir partageable par la génération de liens entre des documents. Il oblige à une vision de la connaissance non plus centrée comme une organisation déductive

d'informations et de savoir-faire, mais comme un travail d'association entre des problèmes et des notions générales et les relations que l'on peut tisser entre toutes ces notions. Pour un enseignant, habitué aux discours et aux progressions, cela oblige à passer d'une vision linéaire d'exposition de connaissances à un réseau de relations facilement accessibles à un utilisateur débutant. Il faut arriver à formaliser un savoir opératoire pour les débutants sans le contraindre à subir une trop grande succession d'étapes.

Avec un groupe d'instituteurs, nous avons ainsi essayé de bâtir un noyau d'une grammaire accessible en ligne. L'idée était de concevoir une grammaire en hypertexte directement consultable à partir d'un traitement de textes et permettant le plus aisément possible de corriger ses fautes en appliquant les règles fournies au contexte courant. Ce travail n'a pu être mené à terme mais a suscité un certain nombre de réflexions :

- la difficulté de la tâche a tout d'abord permis de mieux comprendre les obstacles rencontrés par les élèves : l'organisation globale du domaine est problématique, les nomenclatures officielles sont peu adaptées.
- minimiser la tâche de l'utilisateur en lui fournissant le plus rapidement possible les informations pertinentes, sans disposer d'un analyseur syntaxique, mais en s'appuyant uniquement sur les caractéristiques des mots écrits impose d'inclure une forme de diagnostic (les types d'erreurs les plus courantes) dans la reconnaissance de la demande. De plus, il faut prévoir toutes les informations susceptibles d'être intéressantes dans un contexte donné.
- le travail en collaboration est encore une tâche inhabituelle, bien qu'elle semble de plus en plus nécessaire (voir collaboration). (démonstration accessible en annexe 3, C.e, page 234)

Ces difficultés montrent que les nouvelles techniques vont introduire des changements très profonds dans la conception même et l'organisation des connaissances. Il n'est plus possible de disposer de toutes les informations nécessaires pour résoudre un problème réel, par contre, il est impératif de savoir comment y accéder et de maîtriser ces modes d'accès. L'accumulation d'informations relève du remplissage du tonneau des Danaïdes!

Des travaux ont été poursuivis sur des domaines plus simples, hors du cadre de l'aide en ligne pour supprimer le problème redoutable des modes d'accès contextuels. Ainsi, un compte rendu d'activité d'une classe transplantée a pu être mené à terme, le travail en collaboration ayant pu s'établir sans difficulté majeure. Il faut noter que la simplicité de la création des liens par simple marquage des mots évite considérablement les problèmes techniques. Un travail analogue est en cours en épistémologie de l'histoire avec J. M. Baldner.

Une caractéristique de tous ces systèmes est d'être conçus pour l'utilisateur et de lui simplifier le travail. Cette simplicité se retrouve dans la création. Cette dernière est d'ailleurs de type analogique, non logique, i. e. n'oblige à passer par une phase de formalisation dans un langage intermédiaire. On définit des zones ou des liens directement et on peut tout de suite se mettre à la place du 'lecteur'. Il n'y a pas de programmation à faire, juste un système de marquage : la frontière entre auteur et utilisateur n'est pas facile à délimiter. L'analyse locale peut être cependant difficile, il faut renoncer à avoir une vision détaillée complète du système. En particulier, les représentations planes du type organigramme ne peuvent être que partielles (sinon ce n'est plus de l'hypertexte). Les modes de pensée associés ne sont pas séquentiels, et sont développés par l'usage de langages évolués dépassant le cadre simpliste du pas à pas (voir les environnements Logo : PROD, PROLOGO, MULTI-TORTUES, voir II.1.b.3, page 35).

On peut reprendre une analogie avec ARRIA : la grammaire n'est pas décrite par une déclaration exhaustive de formes admises, mais par des contraintes reliant les connecteurs, les fragments et leurs types.

Chapitre III.3

Problèmes d'usage et impact sur l'éducation

III.3.a Problèmes d'usage des hypertextes

" Though we recognize the power of hypertext, our enthusiasm is tempered by the limitations of one part of the system - the user. We can improve the tool, the computing machine itself, as well as the working environment, but we cannot escape the constraints of human information processing. " [HERRSTROM & MASSEY 89]

III.3.a.1 Problèmes de parcours ('browser')

Les hypertextes n'échappent pas aux difficultés classiquement rencontrées dans l'usage des technologies informatiques. Ces obstacles proviennent partiellement d'une maîtrise incomplète des concepteurs et d'une ignorance des modes d'usage de cet outil encore très neuf. *« La richesse des représentations non linéaires amène un risque potentiel d'indigestion intellectuelle, de perte du but poursuivi et d'entropie cognitive »* [DEDE 88].

Un hypertexte mal conçu et mal structuré peut avoir un effet délétère, les utilisateurs pouvant se perdre en essayant de suivre un cheminement confus qui semble totalement aléatoire. Ainsi, Foss [FOSS 88] classifie les principales difficultés d'orientation :

- la désorientation proprement dite, due à l'ignorance de la structure du réseau et de la position courante,
- la gestion de la tâche rendue complexe par l'absence de stratégie cohérente de parcours et de trop nombreuses digressions, ce qui conduit à une surcharge cognitive ('cognitive overload') due à l'effort et à la concentration nécessaires pour maintenir différents travaux ou chemins en même temps,
- le manque d'indices discursifs, ces derniers étant généralement présents dans les textes traditionnels).

Ces problèmes amènent souvent une perte de contexte, le parcours dans l'hypertexte se transforme en un collage d'informations contingentes qui exclue la compréhension.

"This tangle of linkages becomes an even more critical limitation to the user when navigation in hypertext is unmediated by an instructor or experienced guide. A user may just glance over the surface of a body of knowledge without integrating it into a personal knowing. And if pathways are too firmly established, then the point of hypertext is lost." [BARRETT 88] (à relier au risque de perte de contexte dans l'usage des heuristiques, voir VI.2.b).

A ces problèmes, Dede [DEDE 88] ajoute les risques d'explosion combinatoire (analogue à un système de diagnostic) et les dysfonctionnements de la communication collective (pour les hypertextes incluant la collaboration). On peut y inclure certaines difficultés propres à l'interface ou au mode de représentation utilisé. Ainsi la facilitation initiale des interfaces graphiques conduit à une forme de communication encore peu usuelle. Les utilisateurs doivent savoir lire une information visuelle et établir des liens entre le verbal et le visuel. Cette grammaire, la façon de combiner les informations graphiques, n'est pas si intuitive qu'on le croit [RUBENS 89].

Une première réponse à ces problèmes consiste à fournir diverses aides à la navigation. Ainsi J. F. Rouet [ROUET 90] suggère qu'un *« aménagement de l'interface (par l'isolement spatio-temporel des différents types d'information, par exemple) pourrait favoriser l'orientation locale des processus de traitement »*.

En contrepoint des remarques précédentes, on peut cependant s'interroger sur la complexité réelle des difficultés soulevées, et plus particulièrement le problème de la désorientation.

G. Landow [LANDOW 89a] remarque que beaucoup d'auteurs qui pensent que la navigation reste un problème non résolu suggèrent deux types de solution : l'utilisation de cartes globales et le recours à l'intelligence artificielle. Il poursuit en soulignant que la première solution n'est satisfaisante que pour de petits corpus tandis que la seconde n'est pas encore réellement opérationnelle. Néanmoins, son expérience d'Intermedia suggère que la navigation et l'orientation ne sont pas des problèmes majeurs.

La position de G. Landow est corroborée par d'autres expériences [LEGGET & Al. 90]. De même, nous n'avons pas rencontré ce type de problème dans les différents logiciels intégrant des hypertextes présentés au chapitre précédent (voir III.2, page 91). Il faut noter que ces hypertextes sont très structurés et ne conduisent pas à des recherches très profondes. De plus, les accès sont filtrés par les point de vue (LYRE) ou par le contexte (HyperInfo). La tâche à effectuer est structurante et donne un sens global à la recherche ce qui diminue considérablement les risques de 'noyade'. Dans l'utilisation d'Apilog, c'est plus une incompréhension sur la finalité de l'hypertexte intégré qui pose problème, ainsi qu'une ergonomie déficiente.

On peut même adopter une position plus fondamentale, en considérant que le fait de surmonter la désorientation incombe à l'utilisateur, c'est un obstacle qu'il doit franchir pour accéder à une bonne connaissance du domaine présenté. *" Disorientation can be a good thing for learning. ... While conventional aids to navigation in hypertext are necessary for readers to find their way around hyperspace, we concluded that they were of little value in the more fundamental navigation for learning, that of conceptual space. However, getting lost in the latter sense may not be whether a map can be found but whether one can be created by the learner."* [MAYES & Al. 90] On retrouve ici l'idée de l'erreur comme moteur de l'apprentissage (voir II.1.b.2.a, page 33, sur Logo et la remarque de SLATIN, III.2.b.3.a, page 95). En pratique, il est cependant difficile de dissocier ces deux types de désorientation (voir V.1.d.2, erreurs de surface et

erreurs profondes).

Pour résumer, une part importante des difficultés rencontrées semble provenir d'hypertextes mal construits et d'interfaces inadaptées, ce qui est somme toute normal dans l'utilisation d'un medium récent et encore mal connu. Il ne faut pas négliger non plus le fait que la maîtrise de cet outil réclame des aptitudes nouvelles et spécifiques que les utilisateurs n'ont pas encore développées, aptitudes pour la navigation, l'intégration d'informations, etc. De nouvelles stratégies sont nécessaires, elles devraient naître et croître avec l'usage grandissant des hypertextes. On peut considérer que l'usage de la photocopieuse est d'une certaine façon préparatoire à l'hypertexte, dans la duplication de ressources pour les intégrer à d'autres travaux.

Un dernier obstacle, lui aussi classique, est lié à des problèmes culturels. L'acceptabilité globale d'un système informatique est une combinaison de son acceptabilité sociale et de son acceptabilité pratique [NIELSEN 90]. Pour illustrer cette assertion, J. Nielsen prend l'exemple de LYRE (III.2.b.3, page 95). Le fait que l'élève ne puisse lui-même choisir ses points de vue n'est pas acceptable dans les pays scandinaves, car cela limite les possibilités de découverte indépendante. D'un autre côté, il comprend que la tradition scolaire française ne laisse pas cette possibilité qui pourrait être rejetée comme restreignant l'autorité de l'enseignant. Ceci ouvre un débat sur les modes d'accès et d'intervention permis aux utilisateurs et aux protections éventuelles intégrées dans les systèmes. Dans le cadre scolaire formel, faut-il permettre à l'élève de faire des erreurs ou de suivre des pistes sans objet, faut-il tout restreindre? (voir IV.2.c, page 129).

Enfin, à côté des débats théoriques, il faut intégrer les imperfections inévitables des systèmes informatiques : aucun d'entre eux n'est complètement dépourvu de bogues, et ces bogues auront certainement un impact sur les utilisateurs [NIELSEN 90]. On développe ainsi des stratégies pour 'vivre avec', en effectuant par exemple des sauvegardes fréquentes, en inventant des tactiques locales pour contourner ceux qui sont répertoriés, etc. Ces imperfections se rajoutent à l'incomplétude structurelle des systèmes informatiques (voir IV.2.d, page 132 et III.3.b, page 107).

III.3.a.2 Problèmes de création

Dans la création d'un hypertexte, on distingue deux approches complémentaires :

- une approche déductive qui nécessite de commencer avec une structure du contenu ou la connaissance d'un expert du domaine traité,
- une approche inductive qui s'appuie sur les 'patterns' d'accès vérifiés auprès des utilisateurs.

On peut distinguer aussi trois niveaux correspondant au type d'utilisation prévu :

- utilisation personnelle : ce cadre n'impose pas spécialement de démarche, c'est celui qui introduit l'information qui sera amené à la rechercher, les défauts d'organisation pourront être compensés par des procédures personnelles de parcours très efficaces (une forme de compensation des 'bugs'). Cette forme de développement, favorisé par des outils très simples d'emploi, produit des systèmes quasiment inutilisables pour toute autre personne que le concepteur, les risques de désorientation étant ici maximum!

- utilisation restreinte, dans un milieu proche du concepteur : la connaissance préalable des utilisateurs et le recours possible au concepteur en cas de problème simplifie les problèmes de développement. L'outil est conçu en adaptation au cadre d'utilisation prévu et demeure peu exportable.
- utilisation générale : on tombe dans un cadre de diffusion nécessitant un travail plus complexe, intégrant les approches déductives et inductives mentionnées plus haut.

En fait, la création d'un hypertexte impose le réexamen complet des connaissances d'un domaine : il faut pouvoir passer d'un mode séquentiel d'exposition des connaissances, à un mode partagé ou distribué où il faut faire apparaître les liens entre les diverses connaissances utiles sans imposer un cheminement préétabli. Il faut intégrer de plus de nouvelles conventions qui ne sont pas encore bien établies.

"Authors of hypertext materials must develop and employ stylistic and rhetorical devices that extend beyond those associated with writing texts for publication as a printed book. In so doing, they must distinguish carefully between adapting printed works to hypertext and creating new ones in this new information medium." [LANDOW 90]

Les règles d'or de Shneidermann (voir III.1.a.3.d, page 84) donnent des conseils méthodologiques généraux, mais la rhétorique de l'hypertexte reste néanmoins en grande partie à inventer.

L'une des problématiques actuelle est celle de l'automatisation des liens hypertextes, ou celle de l'utilisation conjointe d'autres techniques (bases de données, interrogation 'full text', intelligence artificielle, etc.).

III.3.a.3 Systèmes d'aide

Le cas des systèmes d'aide et de documentation en ligne présente certaines particularités, du fait de leur cadre d'utilisation spécifique. En effet, les concepteurs doivent centrer des utilisateurs spécifiques sur les seules informations nécessaires pour accomplir des tâches informatisées bien définies, connaissant leurs aptitudes [HERRSTROM & MASSEY 89]. Ainsi, le besoin fondamental de l'utilisateur consiste en des procédures explicites et des informations de référence accessibles dans des chemins prévus. De plus, les modes d'usage habituels introduisent des complexités supplémentaires. Le comportement type d'un nouvel utilisateur consiste à essayer une action ou taper une commande d'une précédente version du logiciel avant d'examiner la documentation pour les modifications. C'est l'erreur ou le blocage qui conduit à la lecture de la documentation. Le point central est le conflit entre les attentes et les 'réponses'. Les hommes attendent des informations aussi pertinentes et adaptées de la machine que s'il s'agissait d'un humain. Si cette attente n'est pas satisfaite, ils perdent leur confiance dans la machine [RUBENS 89].

Les quelques tests effectués en formation avec des systèmes d'aide (voir III.2.c, page 99) montrent que les utilisateurs les moins avancés rechignent à s'en servir. Ils préfèrent poser directement les questions à l'enseignant présent qui est plus apte à interpréter leur demande. Les utilisateurs plus 'débrouillés' par contre explorent rapidement l'outil intégré et sont opérationnels pour trouver ou retrouver l'information dont ils ont besoin. Dans d'autres contextes (utilisation d'EMACS), certains utilisateurs vérifient auprès de l'enseignant les précisions fournies par le système d'aide en ligne. Certains problèmes correspondent d'ailleurs à des échecs répétés dans l'usage de la documentation écrite.

Un entraînement spécifique à la recherche d'informations dans un hypertexte devrait peut-être être mené pour donner confiance aux utilisateurs hésitants et leur prouver que l'outil peut leur fournir une aide effective. On a trop tendance à sous-estimer la difficulté des premiers pas, et à croire que la simplicité de parcours implique automatiquement une simplicité de recherche (voir V.1.b.3).

Les systèmes d'aide intelligents sont une voie de recherche intéressante, prenant en compte le profil (ou un modèle) de l'utilisateur et son chemin. Une idée standard est de remplacer les liens hypertextes statiques par des systèmes de règles prenant en compte ces éléments. On ne pourra cependant pas supprimer les problèmes liés aux modes d'usage et éviter de se heurter à une forme d'incomplétude obligeant à transférer une partie du problème du côté de l'utilisateur.

III.3.b Hypertexte et Education

" Although hypermedia promises great potential for instruction, its efficacy is neither established nor without likely problems. Hypermedia learning systems will place more responsibility on the learner for accessing, sequencing and deriving meaning from information. This added responsibility will entail added cognitive processing requirements on the learners. In many ways, this increased processing load appears consistent with constructive conceptions of learning and therefore desirable. " [JONASSEN & GRABINGER 90]

L'hypertexte est avant tout un outil, une ressource complémentaire, qui n'est pas destinée à supplanter, mais plutôt compléter d'autres outils (plus un outil d'apprentissage qu'un outil d'enseignement). A ce titre, il est impératif que l'enseignant intègre son utilisation dans son cours. Cette optique amène d'ailleurs rapidement à une évolution du rôle même de l'enseignant.

G. Landow [LANDOW 89b] décrit ainsi son expérience de professeur dans l'utilisation de 'Context 32' (voir INTERMEDIA, III.1.a.2.d, page 80) : *" my role of instructor becomes more that of coach than teacher. Furthermore, as students contribute increasingly to the corpus of documents in Context 32, the distance continues to narrow between teacher and student, author and reader, designer and user. "* (voir Schoenfeld, VI.2.c)

On peut retrouver un cycle décontextualisation/contextualisation qu'on retrouve dans les processus d'apprentissage. Les éléments d'information dans l'hypertexte sont décontextualisés, ils sont recontextualisés dans un cheminement donné, et sont de nouveau décontextualisés dans un cadre de généralisation (même idée dans la résolution de problèmes, méthode locale dans un contexte donné, décontextualisation pour la généralisation, voir VI.3.b, page 194).

Au niveau de la motivation ou des situations permettant de faciliter l'apprentissage, il faut noter qu'elles sont en dehors de l'hypertexte lui-même [WHALLEY 90].

Il est légitime aussi de s'interroger sur les limites d'un tel outil et de préciser les types d'activité permis à l'élève.

(Cette question a déjà été évoquée, voir III.3.a.1, page 103). L'un des points importants est la création de liens par les élèves. Le fait d'effectuer des liaisons confère cette authenticité qui, bien que contestable, est apte à être prise pour la réalité par un élève naïf [DOLAND 89]. C'est une question encore essentiellement ouverte qui dépend des conceptions pédagogiques

que l'on peut avoir. Il faut noter que dans un cadre d'institutionnalisation, ce problème se pose aussi (dans une communauté collaborant à l'écriture d'un hypertexte).

Peut-on laisser les étudiants établir des liens non pertinents voire absurdes? Peut-on définir un lien absurde? surréaliste? Quelles contraintes internes respecte un outil et comment en rendre conscient l'utilisateur? Doit-il même en être conscient?

Le problème de complétude se pose ici d'une manière duale de celle de l'intelligence artificielle. Le recours à des documents externes est une reconnaissance explicite de la non-omniscience de l'enseignant, qui est censé cependant être capable de faire le lien entre les informations trouvées et le cours qu'il dispense aux élèves (en quelque sorte il dispose de connaissances de niveau supérieur lui permettant d'organiser et de structurer les informations récupérées). Virginia Doland [DOLAND 89] souligne cependant l'impression illusoire de complétude qui peut être créé par les chemins au travers d'un système hypertexte. Bien que l'ensemble de ces chemins peut sembler infini, le réseau de liens est fondamentalement fini et reflète les choix et interprétations du créateur du document. Ainsi, l'incomplétude existe aussi au niveau de l'hypertexte, mais peut-être d'une manière plus pernicieuse, en risquant d'entretenir une idée de complétude mythique qui ne peut en aucun cas exister d'une manière pratique (voir IV.2.d, page 132).

On peut aussi tisser un lien entre la problématique hypertexte, centrée sur l'utilisateur et les tentatives d'explications intégrées aux systèmes experts. L'hypertexte transfert une partie de la difficulté du problème du côté de l'utilisateur, en lui confiant la responsabilité d'accéder par lui-même aux points qu'il a du mal à intégrer. Au niveau curriculum, il faut rappeler que les études montrent que l'utilisateur est un très mauvais décideur au niveau de son propre cursus d'apprentissage, et, en tout cas, inférieur à un enseignant ou à une machine convenablement programmée [ATKINSON 76].

Remarque : Un parallèle intéressant existe entre la problématique hypertexte et la démonstration. L'effet fragmenté d'une démonstration rejoint la construction d'un hypertexte, avec l'existence commune d'éléments de base, i. e. l'absence de régression infinie. La rédaction de la démonstration obéit à une rhétorique particulière, et correspond au parcours d'un chemin dans un graphe ou un hypergraphe. Peut-on et faut-il apprendre les règles de cette rhétorique à partir d'un contenu déjà constitué (la problématique d'ARRIA rejoint directement celle d'un lecteur d'un système hypertexte sans possibilité d'annotation, i. e. contraignant l'utilisateur à suivre uniquement les chemins prévus par l'auteur).

Pour conclure, la généralisation de l'hypertexte devrait conduire à une révolution analogue à celle qui a accompagné le passage du manuscrit à l'imprimerie. Malheureusement, de tels bouleversements sont à la fois difficilement prévisibles et incontrôlables, leurs conséquences ne sont pas encore mesurables. L'impact sur l'enseignant sera sans conteste considérable, mais il risque d'être très (très) progressif.

Quatrième Partie

ARRIA :
Un système ouvert de gestion de fragments pour l'apprentissage de raisonnements et de leur communication

Aide
(au Raisonnement)
et à la Rédaction
Intelligence
Artificielle

La parenthèse est due au fait que ARRIA devait initialement être nommé ARIA, mais le nom étant déjà sélectionné, un R a dû être ajouté! Cette modification n'est pas totalement innocente, puisqu'elle concerne un des points discutables du système : est-ce qu'un travail spécifique sur la rédaction (ou la communication suivant certaines contraintes) d'un raisonnement a un effet sur le raisonnement lui-même.

ARRIA (ou Hyper-Arria) est un système général de développement destiné à aider des apprenants dans des tâches incluant des raisonnements et leur communication. Il s'appuie sur une première réalisation de type scolaire faite sur l'apprentissage de la rédaction de démonstrations mathématiques (ARRIA).

Le type d'application visé amène quelques remarques préliminaires.

Apprendre à raisonner est ardu : il n'existe pas de méthode miracle dans ce domaine. C'est un vrai problème de formation pour lequel le recours à des techniques informatiques consiste à tenter de découvrir de nouvelles voies qui peuvent compléter d'autres techniques plus traditionnelles. Il faut donc prévoir et faciliter une intégration dans des cursus (voir III.3.b, page 107).

Il s'agit d'enseigner des démarches dans des domaines réputés difficiles. Ceci induit une forme d'interaction centrée sur l'activité de l'utilisateur, dans des environnements où ce dernier doit produire quelque chose. On aboutit ainsi à une forme mixte tuteur/assistant.

La machine doit pouvoir résoudre, au moins partiellement, les exercices qui se posent à l'apprenant, ce qui impose la construction de résolveurs ou, à défaut, de ce qu'on pourrait appeler des semi-résolveurs. Ceci exclut l'usage des langages auteurs, ces derniers n'offrant pas les outils suffisants pour de tels développements.

Ces quelques remarques générales seront développées dans la suite de ce texte, mais situent déjà les divers problèmes à aborder et les types de solutions préconisées.

- la partie IV.1 présente le logiciel ARRIA et la problématique initiale ayant conduit à sa réalisation ainsi que l'extension CRE-ARRIA autorisant les formateurs à intégrer leurs propres exercices.
- la partie IV.2 est une réflexion générale qui s'appuie sur les différentes réflexions qui ont pu naître de sa réalisation et des premiers retours d'expérimentation : problèmes de complétude et d'usage dans l'enseignement.
- la partie IV.3 s'attache à décrire un système général d'aide à l'apprentissage de raisonnements qui pourrait être construit à partir de l'architecture conçue pour réaliser ARRIA.

Chapitre IV.1

ARRIA et l'apprentissage de la démonstration mathématique

ARRIA est un système qui a été réalisé à partir d'un scénario primé dans le cadre du premier concours organisé par le Ministère de l'Education Nationale (scénario présenté par G. CAMPAGNE, D. LATAPIE-BRIAN, J. B. MELET et M. THOMAS). J'en ai assuré le développement en collaboration avec l'un des auteurs précédents, J. B. Melet.

L'idée initiale consiste à ramener la rédaction d'une démonstration à la réalisation d'une sorte de puzzle : il faut reconstruire en remettant en ordre un ensemble de fragments fournis dans le désordre.

Cette problématique rejoint certaines recherches en didactique des mathématiques : « *Les tâches spécifiques à une démarche de démonstration sont des tâches d'organisation. Elles supposent que l'on dispose explicitement de tout le corpus des énoncés nécessaires pour la démonstration, c'est-à-dire que l'on dispose non seulement des énoncés de départ et de l'énoncé-résultat mais aussi des règles de substitution à utiliser. A aucun moment, dans une tâche d'organisation déductive on ne doit avoir à chercher lequel des théorèmes déjà appris pourrait servir. Toutes les pièces doivent, en quelque sorte, être déjà sur la table, sans aucune hésitation possible.* » [DUVAL & EGRET 89]

IV.1.a Enseignement de la démonstration

IV.1.a.1 Qu'est-ce qu'une démonstration?

Rappelons les définitions suivantes [BALACHEFF 87] :

- Explication : discours visant à rendre intelligible le caractère de vérité, acquis pour le locuteur, d'une proposition ou d'un résultat.
- Preuve : explication acceptée par une communauté à un moment donné.
- Démonstration : preuve sous la forme d'une suite d'énoncés organisée suivant des règles déterminées (faits plus déductions).

- Raisonnement : activité intellectuelle de production d'information.

Avant de chercher à enseigner à faire des démonstrations, il faut bien sûr pouvoir définir ce qu'est une démonstration. Malheureusement, ce qui semble «tomber sous le sens» n'est finalement pas du tout évident. En effet, la démonstration n'a pas de statut scolaire : le cours de mathématiques ne la définit pas. Face à un texte d'élève, l'enseignant décide si c'est acceptable ou non en ne fournissant que des justifications locales. En gros, l'élève ne sait pas ce qu'est une démonstration tandis que le professeur est capable d'en reconnaître une. Ainsi la démonstration scolaire n'est pas celle du mathématicien, qui s'inscrit dans une communauté de pairs, mais un exercice, dont la finalité reste trop souvent obscure, évalué par une autorité supérieure dans le cadre de contraintes sociales.

Si on se place dans une perspective logique, la démonstration est généralement vue comme une suite de propositions commençant par des prémisses, utilisant des théorèmes et appliquant le modus ponens aux propositions précédentes pour construire une chaîne logique des prémisses aux conclusions. Ce type de définition masque deux aspects importants d'une démonstration :

- sa structure d'arbre,
- le fait que l'on peut travailler aussi bien en avant (des hypothèses vers la conclusion) qu'en arrière (de la conclusion vers les hypothèses) [BURTON 88] [BALACHEFF 78].

Dans une perspective pragmatique, on sépare habituellement la production d'une démonstration en 2 phases :

1. trouver un plan,
2. traduire ce plan dans une preuve effective.

La première phase correspond à la recherche d'un chemin plus ou moins complet reliant les hypothèses et la conclusion, la deuxième étant la traduction de ce chemin dans un langage pseudo-mathématique. Cette dichotomie, si elle reflète bien les moments différents dans la production d'une démonstration, ne doit pas conduire à une distinction abusive du type fond/forme. En effet, on a trop tendance à considérer la première phase comme seule relevant d'une intelligence créative, la deuxième étant une simple transcription mécanique [ANDERSON 83b]. Ceci est sans doute vrai pour ceux qui connaissent parfaitement le rôle et l'écriture de la démonstration, mais sûrement pas pour ceux qui n'ont pas encore attachés de signification claire à cette notion.

Il faut noter que le recours à un programme informatique tel que le Geometry Tutor d'Anderson [ANDERSON & Al. 85]) rend globalement inutile la phase de traduction puisque l'arbre construit à l'écran est une représentation formelle exacte de la démonstration.

Un élève ne dispose généralement pas de ces ressources et est contraint de fournir une copie obéissant à des contraintes relativement strictes de présentation (normes sociales). Ces dernières conduisent ainsi rapidement à des déviations sur la nature même de l'activité. Schoenfeld [SCHOENFELD 87b] souligne ainsi que les élèves concernés par la forme (écrivant par exemple des preuves de géométrie sur deux colonnes avec des abréviations correctes) passeront plus de temps à se tracasser sur la forme de leur réponse qu'à essayer de comprendre le résultat qu'ils sont en train de mettre par écrit. Du point de vue de ces élèves, les problèmes de preuve confirment ce que l'on connaît déjà ou ce qu'on leur a affirmé comme étant vrai. Les problèmes de construction vous demandent de trouver quelque chose. Ainsi, quand ils travaillent sur des problèmes de construction, ils sont en quelque sorte dans un mode de

découverte, les résultats des preuves correspondent à un mode de confirmation et sont donc inapplicables.

On peut résumer les précédentes remarques par une formule lapidaire :

démontrer = convaincre

Dans le contexte scolaire, la démonstration est la trace censée refléter la compréhension du problème par l'élève, trace qui doit être suffisamment convaincante pour que le professeur mette une bonne note.

« La démonstration normalement est faite pour convaincre quelqu'un : se convaincre d'abord soi-même et ensuite convaincre les autres. Mais, dans le dialogue avec le professeur, il n'y a personne à convaincre. Alors, à quoi sert une démonstration? C'est un exercice rituel dont une bonne partie ne comprend pas la signification. » [LACOMBE 88b]

Remarque : ARRIA, traitant de problèmes géométriques niveau seconde, évite le cas tendancieux de certaines démonstrations algébriques où le rappel de la définition tient souvent lieu de preuve. Ainsi, pour prouver qu'une certaine relation R est symétrique, on pourra trouver : oui car pour tout x et tout y, x R y implique y R x sans utilisation de la définition spécifique de la relation R.

IV.1.a.2 Construction et rédaction d'une démonstration

On peut opposer une démonstration «logique» où chaque étape doit être entièrement explicitée à une démonstration «scolaire», ou «pragmatique», dans laquelle des éléments sont sous-entendus. Le problème est de tracer la frontière entre ce qui doit être mentionné et ce qui peut être passé sous silence. Ainsi, quand on résout l'équation $2x = 0$ dans une classe de Terminale, on n'invoque pas la règle sur un produit de facteurs pour déduire que $x = 0$.

En fait, pour savoir ce que l'on peut «oublier», on utilise une règle empirique : peuvent être considérées comme implicites toutes les connaissances normalement acquises dans les classes antérieures. Ceci se justifie d'autant plus dans le cadre scolaire que les démonstrations sont un moyen d'évaluation des connaissances fournies dans le cours : on peut quasiment parler de lien causal «scolaire» entre les démonstrations demandées et les définitions et théorèmes vus quasiment au même moment dans le cours de mathématiques (voir VI.3.b, page 194)

La rédaction de la démonstration s'effectue dans une langue proche du français mais conservant néanmoins certaines particularités, à la fois aux niveaux lexical, syntaxique, sémantique et pragmatique [BALACHEFF 78] [LACOMBE 84] :

- niveau lexical : symboles mathématiques ou logiques;
- niveau syntaxique : on utilise généralement une langue «simple», en évitant par exemple les enchâssements de propositions relatives, en utilisant des phrases courtes, etc.;
- niveau sémantique : pas de figure de style, ambiguïté des connecteurs comme «et», «or»;
- niveau pragmatique : recouvre le problème des implicites qui vient d'être évoqué.

On peut aussi distinguer une rédaction que l'on peut qualifier d'experte, où on ne rementionne pas les résultats que l'on vient de démontrer, et dans laquelle on a tendance à enchâsser les arguments (à l'aide du «or»), et une rédaction plus débutante, avec une maîtrise de la logique moins affirmée, où il vaut mieux éviter le recours à des ellipses.

IV.1.a.3 Comment enseigner la démonstration

« La méthode d'apprentissage de la démonstration à peu près universellement en vigueur aujourd'hui, consiste à montrer à l'élève une ou plusieurs démonstrations et à lui demander de faire «pareil» (éducation par imitation ; voir «psittacisme» dans le Petit Larousse). » [IREM 79]

Cette phrase illustre bien le manque de réflexion en profondeur sur la démonstration et son apprentissage : ne pouvant expliciter réellement ce qu'est une démonstration, la seule ressource est d'exhiber des exemples «typiques» ; aux élèves de généraliser s'ils le peuvent (voir cependant les recherches actuelles sur l'apprentissage de la démonstration et le débat scientifique). Les enseignants, directement confrontés à des élèves en difficulté, ont dû développer diverses techniques pour essayer de les aider, ce qui conduit à faire identifier les constituants d'une démonstration (phrases ou propositions) et s'intéresser aux liens logiques entre ces constituants. Ces démarches d'explicitation sont à la base d'ARRIA.

Si on ne sait pas trop comment faire pour enseigner l'«art» difficile de la démonstration, on peut cependant lister un certain nombre d'incompréhensions, d'erreurs ou de blocages constatés chez les élèves :

- mauvaise lecture de l'énoncé (que faut-il faire?);
- incapacité à mettre en situation les connaissances du cours;
- absence de stratégie de recherche;
- incompréhension globale des constituants de la démonstration : qu'est-ce qu'une hypothèse, une conclusion?
- mauvaise maîtrise de la logique élémentaire;
- incapacité à rédiger, à exprimer un raisonnement par écrit.

Si on considère qu'il y a deux phases ou deux étapes :

1. phase de résolution de problèmes correspondant à la recherche d'un chemin,
2. phase de rédaction, i. e. traduction de ce chemin dans une preuve effective,

Deux questions se posent :

- peut-on faire l'étape 'a' sans comprendre l'étape 'b'?
- la maîtrise de 'b' a-t-elle une influence sur les aptitudes pour 'a'?

Une dernière question est de savoir si la séparation effective de ces deux phases est légitime. Les études réalisées par R. Duval et M. A. Egret apportent des éléments de réponse : *« Tout se passe comme si les élèves qui ont appris à maîtriser le jeu de l'organisation déductive disposaient d'un cadre pour interpréter correctement, pour généraliser des procédures rencontrées et pour contrôler les solutions obtenues. »* [DUVAL & EGRET 89] La séparation des deux phases est nécessaire car ce ne sont pas des tâches de même nature du point de vue cognitif.

IV.1.b EIAO et démonstration

Les travaux sur la démonstration concernent principalement deux domaines :

- la logique (II.2.b.7, page 63),

- la géométrie (II.2.b.6, page 63).

Les réalisations dans le domaine de la géométrie sont soit de type micromonde, soit de type tutoriel bâti sur un démonstrateur.

IV.1.b.1 Micromondes en géométrie

L'extension de la géométrie tortue du Logo traditionnel vers la géométrie euclidienne a déclenché sans conteste un intérêt international. Thompson [THOMPSON 87] décrit ainsi les programmes EUCLID, MOTIONS et LEGO. Une différence essentielle avec les micromondes préalablement étudiés (voir II.1.b.1, page 30), est qu'il n'y a plus d'objet transitionnel : les objets traités avec l'ordinateur sont les objets courants de la géométrie. Il s'agit ainsi plus d'outils permettant de réaliser des figures et de faire des conjectures (des extensions du papier/crayon) que de micromondes. L'abandon progressif d'un langage de commande au profit d'une interface de manipulation directe montre une volonté de simplifier la tâche de l'utilisateur en lui fournissant le moyen de manipuler directement les objets qu'il connaît.

Ainsi, on arrive à des réalisations qui sont des environnements bâtis à partir de Logo concernant la géométrie plane ou la géométrie dans l'espace :

- EUCLIDE [ALLARD & PASCAL 87],
- EUCLIDE 'espagnol' [FABREGA 88], [MONTES 88],
- 'GEOMETRY SUPPOSER' [YERUSHALMY 88],
- EUCLIDE bulgare [SENDOV & DICHEVA 88],
- [FLORIS & BEVACQUA 89] système auteur pour la géométrie.

Remarque : l'éloignement de la référence aux micromondes les rapproche plus de langages auteur spécialisés sur la géométrie avec lesquels des séquences balisées peuvent être construites (voir IV.2.c.4, page 131).

CABRI [BAULAC 90] s'éloigne de la référence à Logo en intégrant une interface de manipulation directe permettant de déformer dynamiquement les figures.

IV.1.b.2 Les démonstrateurs

Divers démonstrateurs sur le domaine de la géométrie ont été construits pour être intégrés à des tuteurs (voir II.2.b.6). Les Actes de l'Université d'Eté de Toulouse de septembre 90 permettent de faire le point sur ce sujet.

Il faut noter que le système ARRIA se démarque en abordant plus le problème de la rédaction que celui de la recherche de la solution.

IV.1.c Architecture d'ARRIA

ARRIA est la rencontre entre des préoccupations d'enseignants de mathématiques ayant imaginé un scénario et le système de développement SEVE (III.2.b.1, page 92) dont les caractéristiques (association d'hypertexte et de Prolog) permettaient un développement rapide.

IV.1.c.1 Le scénario

Comme il a déjà été signalé plus haut, le scénario a été bâti autour de l'idée de puzzle. Une démonstration correcte est décomposée en fragments élémentaires qui sont proposés en vrac à l'élève. Ce dernier doit reconstruire une démonstration en utilisant ces fragments et des conjonctions ou des signes de ponctuation. Le déroulement s'effectue en trois étapes (voir en annexe le déroulement d'une session, D.b, page 242) :

1. Lecture active de l'énoncé avec diverses aides accessibles. En particulier, un énoncé «bis» donne des indications stratégiques sur une démonstration possible.
2. Typage des fragments proposés, avec quatre possibilités :
   - hypothèse ou donnée;
   - conclusion;
   - outil (théorème, résultat du cours);
   - résultat intermédiaire.
3. Rédaction effective en utilisant les fragments précédents, les conjonctions «et», «ou», «or», «car», «donc» («mais» et «ni» ont été supprimés), la virgule, le point-virgule et le point. La démonstration une fois réalisée peut être imprimée.

Ce scénario précisait bien le travail devant être effectué par l'élève, mais souffrait d'une grosse lacune en ne donnant aucune précision sur la manière dont on pouvait vérifier une rédaction quelconque proposée par un élève. La résolution de ce problème a constitué la partie la plus difficile dans la réalisation d'ARRIA.

IV.1.c.2 Les problèmes de conception

La réalisation des deux premières phases du scénario (lecture de l'énoncé, typage des fragments) ne présentait pas de difficulté, l'hypertexte étant d'ailleurs la solution informatique la plus naturelle pour une interaction basée sur une forme de lecture «à trois dimensions» et la conception de messages d'erreur (voir III.2.a). La vérification de la rédaction était entièrement à faire; nous allons tenter ici de la détailler (voir aussi en annexe, D.d, page 248).

On retrouve tout de suite une opposition logique/pragmatique, correspondant à deux démarches :

- celle de l'enseignant : peu d'exigence de formalisation, pas de critère général d'acceptabilité mais un souci de bonne lisibilité;
- celle de l'informaticien (logicien) : recherche de démonstrations formelles et d'un processus automatisable.

Ceci conduit à préciser l'emploi des connecteurs, et plus particulièrement le «et», le «ou» et le «or», à ne pas admettre trop d'ambiguïtés, et à traiter certains outils comme implicites (nécessaires dans une démonstration logique mais facultatifs dans la rédaction). On arrive à la définition d'une grammaire, qui est essentiellement un compromis entre le langage naturel appauvri, et la logique formelle.

Ainsi, par exemple, le tableau suivant donne toutes les rédactions admissibles pour montrer le résultat R à l'aide de l'hypothèse H et de l'outil O.

Outil non implicite				
H	[et , or]	O	donc	R
O	[et , or]	H	donc	R
R	car	H	[et , or]	O
R	car	O	[et , or]	H
H	donc	R	car	O
O	donc	R	car	H

Outil implicite		
H	donc	R
R	car	H

[et , or] indique que l'on peut choisir n'importe lequel des 3.

La rédaction est vue comme une suite de démonstrations élémentaires, prouvant chacune un résultat intermédiaire nécessaire pour amener la conclusion. On introduit la notion de bloc élémentaire comme étant formé d'un résultat, des fragments intervenant dans sa preuve et des connecteurs utilisés entre chacun de ces fragments. Définir la grammaire revient alors à spécifier toutes les écritures admissibles dans un bloc élémentaire, et les liaisons entre les différents blocs. Les vérifications à effectuer sont ainsi :

- soit internes à un bloc,
- soit en fin de bloc (en liaison on non avec le bloc suivant).

Il faut noter qu'elles sont liées à deux impératifs contradictoires (du fait que toute tolérance et toute ambiguïté rendent l'analyse de l'erreur plus complexe) :

- accepter «toutes» les rédactions correctes (admissibles),
- détecter le plus finement possible les erreurs en cours de rédaction.

En effet, il ne s'agit pas de créer un générateur de rédactions, mais un vérificateur donnant un diagnostic cohérent aux divers écarts constatés. Nous reviendrons plus loin (IV.1.c.5, page 121) sur cet épineux problème qui nécessite des compromis qui peuvent être jugés discutables.

Voici quelques choix d'utilisation des connecteurs :

et Il relie deux fragments de même nature, et plus généralement deux résultats connus (hypothèse, outil ou résultat intermédiaire précédemment démontré). Cependant l'écriture H donc R1 et R2 suppose que H permet de déduire R1 et R2, non que R1 implique R2. Dans ARRIA, on n'utilise pas le connecteur «et» pour enchaîner des résultats se déduisant l'un de l'autre.

or Il relie deux résultats connus, mais pas deux hypothèses consécutives. On n'écrira pas H1 or H2 (mais «et» ou «,»). En effet, l'emploi de «or» met en exergue le fragment qui le suit et ne doit pas séparer deux fragments de même nature.

ou Dans les démonstrations traitées, il n'y a pas de preuve par disjonction de cas, le «ou» n'est donc jamais utilisé dans son sens le plus courant. Il intervient dans ARRIA pour enchaîner deux résultats se déduisant directement l'un de l'autre, sans référence à un outil. Il correspond à l'indication d'une forme de réécriture, et peut toujours être remplacé par «donc». (Ce choix très discutable est souvent très contesté par les enseignants, certains préféreraient la suppression complète du «ou»).

Remarque : le scénario initial reprenait l'ensemble des conjonctions : mais ou et donc or ni car. Les conjonctions «mais» et «ni» ont de suite été écartées.

IV.1.c.3 Le travail effectué par l'élève

Pour rédiger la démonstration, on suppose que l'élève a au préalable effectué un premier travail sur papier en notant une sorte de plan (on rappelle que la phase de recherche n'est pas l'objectif prioritaire du logiciel). Il va alors sélectionner dans des menus un à un les éléments constituant sa preuve. Il choisit alternativement un fragment puis un connecteur, chaque choix étant vérifié directement par le programme.

La sélection d'un fragment s'effectue en deux étapes :

- choix d'un type (hypothèse, outil, résultat déjà obtenu, résultat à démontrer),
- puis, si l'option est validée par le système, sélection d'un fragment du type choisi.

Pour des motivations pédagogiques, on a introduit une vérification en pas à pas de la rédaction de l'élève avec interruption immédiate en cas d'erreur. Ceci permet un guidage constant en fournissant le plus souvent des messages précis et bien adaptés à la situation. Il est d'ailleurs inutile de revenir en arrière, car le système ne valide que des choix conduisant à une solution correcte.

Quelques limitations :

- L'idée puzzle sous-entend une information complète : on dispose de tous les éléments utiles (avec peut-être des éléments superflus), on se trouve dans un univers clos. Ainsi, il n'est pas possible de développer des démonstrations avec des fragments non fournis, ou en les associant d'une manière non prévue par l'auteur.
- Le choix des connecteurs peut être jugé pauvre et producteur de rédactions stéréotypées. Il évite cependant l'écueil d'une croyance un peu naïve à une compréhension de la part de la machine et insiste mieux sur l'aspect mécanique de la rédaction.
- La démonstration en arrière (en partant de la conclusion) n'est pas possible. En effet, la vérification choisie interdit de se servir d'un résultat non encore démontré.
- L'enchaînement de deux connecteurs (ou signe de ponctuation suivi d'un connecteur) n'est pas prévu.

Ces remarques renforcent l'idée d'une prise en compte d'ARRIA dans une progression décidée par un professeur, qui peut intervenir et apporter des compléments sur les points faibles du logiciel. En particulier, rien n'empêche de reprendre les rédactions effectuées pour les réécrire ou les enrichir (en les récupérant sous traitement de texte).

IV.1.c.4 La représentation interne

On peut tout d'abord séparer les modules généraux de ceux qui sont spécifiques à un exercice.

Le programme général se compose :

- de divers modules de gestion du système;
- d'un hypertexte contenant un grand nombre d'informations sur le domaine (géométrie de la classe de seconde) et les messages d'erreur généraux;
- du programme de vérification de la rédaction, écrit en Prolog.

Un exercice comprend :

- la déclaration de l'énoncé, avec éventuellement une figure illustrative (montrant plusieurs configurations différentes);
- un hypertexte contenant des messages d'erreur spécifiques (en particulier pour les erreurs sémantiques correspondant à des mauvaises associations de fragments) et des indications complémentaires;
- un ensemble de faits Prolog caractérisant l'ensemble des fragments et leurs relations.

C'est cette base de faits qui est utilisée par le programme de vérification. L'annexe D.a présente ainsi deux exercices avec leur énoncé, la liste des fragments avec leur type et le paquet de clauses «prouve» qui correspond à l'arbre de preuve. La clause prouve(2,[4,8]) signifie que le fragment 2 se démontre à l'aide des fragments 4 et 8. Le fait : frag(2,r,"les droites (AC) et (BD) sont perpendiculaires") indique que le fragment 2 est un résultat intermédiaire et donne la chaîne de caractères qui lui correspond. La relation de preuve liant les fragments 2, 4 et 8 permet de constituer les blocs élémentaires de la rédaction.

Les erreurs de rédaction détectées sont soit logiques soit sémantiques. Les erreurs logiques correspondent à des mauvais usages des connecteurs ou des types de fragment. Par exemple, faire suivre une hypothèse par un «car». Les messages associés sont généraux puisqu'ils ne prennent pas en compte les sens des fragments manipulés (cela ne joue que sur la forme). Les erreurs sémantiques proviennent de mauvaises associations de fragments. Par exemple, essayer de prouver le fragment X avec le fragment Y alors qu'ils n'ont aucun rapport entre eux. Il n'y a qu'un message général associé à ce type d'erreur, mais on peut en ajouter des spécifiques. Il n'est pas question de prévoir toutes les liaisons erronées possibles, mais d'en rechercher les plus caractéristiques pour fournir le message le plus apte à aider l'élève confronté à ce type d'erreur.

IV.1.c.5 Problèmes de guidage

Très souvent, l'élève n'ayant qu'une idée très vague de ce qu'il faut faire, a du mal à démarrer. Quelles indications doit-on lui fournir pour débloquer cette situation et lui permettre d'avancer? De manière plus précise, quel résultat doit-il d'abord démontrer? ARRIA ne traite qu'en partie ce problème. Les diverses aides autour de l'énoncé fournissent déjà des pistes précises pour la démonstration. Quand l'élève commence la rédaction, les fragments acceptés par le programme lui indiquent qu'il est sur la bonne voie. Quand on suppose qu'il va trop vite, ARRIA lui propose un résultat intermédiaire à montrer.

Par exemple, dans la situation de l'exercice 1 (voir D.a, page 241), avec les deux relations de preuve suivantes :

```
prouve(2,[4,8])
prouve(9,[2])
```

Le fragment 9 se déduit directement du 2, on pourra donc lui indiquer qu'il faut d'abord prouver 2 avant de prouver 9, si 2 n'est pas encore établi. Il faut noter que l'on est incapable de savoir si l'élève a compris, mais a sauté une étape (dans ce cas, il n'y a pas vraiment d'erreur), s'il est un peu perdu ou s'il a fait une simple erreur de manipulation. Dans le doute, le message le plus neutre s'adapte mieux à ces diverses situations.

Si on regarde la coupe de l'arbre de preuve limitée aux résultats à établir, on distingue des chemins très différents. A l'exercice 1 est associée une progression linéaire : `7 -> 8 -> 2 -> 9`.

Par contre, l'exercice 2 a une structure beaucoup plus élargie :

$$\left.\begin{array}{r} 9 -> 1 \\ 5 \\ 7 -> 3 \end{array}\right] -> 6$$

On peut d'ailleurs dégager une notion de bloc (voir [BALACHEFF 78]), un bloc groupant un ou plusieurs blocs élémentaires reliés linéairement. Ainsi, à l'exercice 2 seraient associés 4 blocs (1, 5, 3 et 6). Ceci permet de poser un problème de recherche intéressant qui est de dégager des liens entre cette répartition en blocs et un guidage stratégique de l'élève. Plus généralement, il faudrait voir si on peut classifier les arbres de démonstration et leur associer automatiquement des modes d'intervention. (En particulier, il semble utile d'ajouter des descripteurs aux blocs, comme la précédence, pour mieux rendre compte du cheminement de certaines démonstrations)

IV.1.c.6 Problèmes liés à la rédaction

Nous avons déjà évoqué la distinction à faire entre des rédactions «expertes» où les ellipses et les sous-entendus sont fréquents et les démonstrations «débutantes» où certaines contraintes permettent de mieux clarifier l'articulation logique du raisonnement. Pour améliorer ARRIA, nous avons introduit la possibilité de choisir divers modes de vérification, donc de modifier la grammaire utilisée. Ceci concerne exclusivement les vérifications effectuées en fin de bloc (avec enchâssement sur le bloc suivant).

L'exemple suivant permet de montrer divers types de rédaction possibles.

```
prouve(R1,[H1,O]).
prouve(R2,[R1]).
prouve(R3,[R1,H2]).
```

Quelques rédactions :

H1 et O donc R1. R1 donc R2.	
H1 et O donc R1 donc R2	(pas de rappel de R1)
H1 donc R1 car O donc R2	(pas de rappel de R1 plus éloigné)
H1 et O donc R1 or H2 donc R3	(enchâssement avec le «or»)
H1 et O donc R1 donc R3 car H2	

L'absence de parenthèses conduit à des écritures plus complexes :

H1 donc R1 car O or H2 donc R3
R1 car H1 et O or H2 donc R3

On pourrait généraliser en mettant bout à bout les résultats à montrer dans une phrase (correspondant à un bloc).

IV.1.d Le générateur Cré-Arria

IV.1.d.1 Présentation générale

La demande de nombreux enseignants face à un outil logiciel permettant un travail à partir de quelques exercices (15 dans ARRIA) est de pouvoir ajouter rapidement leurs propres exercices. Ceci autorise une meilleure intégration du logiciel à la progression d'une classe donnée et donc favorise l'appropriation du produit par les utilisateurs.

Cette démarche, pour être efficace, amène plusieurs contraintes :

- fournir un générateur très simple d'emploi. On ne peut limiter l'utilisation d'un tel outil aux seuls formateurs ayant des connaissances importantes en informatiques,
- permettre une création rapide. L'entrée d'un exercice ne doit pas dépasser un quart d'heure (le temps de préparation n'étant pas comptabilisé),
- faciliter les tests et les corrections. Des va-et-vient entre le mode élève et le mode création sont souvent nécessaires; ils ne doivent pas surcharger inutilement le travail,
- offrir un outil fiable capable de vérifier la cohérence des données fournies par l'enseignant pour éviter des surprises désagréables durant l'utilisation en mode élève.

Ces différentes remarques ont été prises en compte dans la réalisation du générateur Cré-Arria, ainsi que certains défauts constatés dans les premières expérimentations d'ARRIA (meilleure ergonomie, plus de souplesse dans la grammaire, des extensions des possibilités d'impression). L'utilisation du mode création s'adresse en priorité aux enseignants, mais sa simplicité de mise en oeuvre permet de faire faire des réalisations aux élèves (une autre approche de la démonstration avec des déclarations locales de liens entre fragments).

La tâche se limite à déclarer les éléments associés à un exercice :

- l'énoncé du problème,
- la liste des fragments avec leur type,
- les relations de preuve entre les fragments.

On dispose pour cela d'un éditeur pleine page structuré, et on peut appeler un système de vérification pour éviter les omissions :

- un résultat intermédiaire sans preuve associée,
- des cycles dans l'arbre de preuve (1 prouve 2 et 2 prouve 1).

Des figures peuvent aussi être intégrées.

Ce générateur permet d'explorer d'autres domaines mathématiques, et donne des éléments pour généraliser les types de raisonnement utilisés dans ARRIA.

IV.1.d.2 Spécificités de la nouvelle version utilisateur

Au module de création est associée une nouvelle version du module utilisateur apportant diverses améliorations par rapport à l'implantation précédente :

- intégration de processus externes,

- 4 grammaires,
- 3 types d'aide automatique,
- sauvegarde/ impression enjolivée (reprise possible sous traitement de texte),
- suppression de l'hypertexte sur le domaine géométrique (il peut être créé par l'enseignant à l'aide d'un outil comme HyperInfo (III.2.c, page 99)).

IV.1.d.2.a Plusieurs grammaires :

Dans ce logiciel, toutes les démonstrations a priori envisageables ne sont pas permises. En effet, la résolution d'un problème aussi complexe ne semble pas encore atteignable avec les machines actuelles. De plus, certaines contraintes d'écriture ont des vertus pédagogiques en forçant les apprenants à déclarer le plus complètement possible les étapes de la rédaction de leur démonstration. Cela permet aussi de mieux suivre leur raisonnement et donc d'être à même de détecter les incompréhensions et les erreurs, et à tenter d'y remédier en intervenant directement au cours de la rédaction. Les contraintes choisies sont obligatoirement en partie arbitraires, mais l'utilisateur doit s'y plier. Ceci peut paraître frustrant à un utilisateur compétent, puisque certains raccourcis ou procédés de style pourront être refusés. Pour adoucir cette contrainte et permettre une progressivité dans l'apprentissage, Cre-Arria offre 4 modes distincts de vérification, du plus contraint vers le moins contraint, ou du débutant vers l'expert.

Dans tous les cas, l'utilisateur est forcé d'aller des hypothèses vers la conclusion, cette dernière étant obligatoirement le dernier résultat à montrer. Un résultat ne peut être montré que si tous les élément intervenant dans sa preuve sont connus, c'est-à-dire si ce sont soit des hypothèses ou des outils, soit des résultats déjà obtenus. Ainsi, Cre-Arria ne permet pas de rédiger des démonstrations en chaînage arrière (en partant de la conclusion), afin de contrôler en pas à pas toutes les étapes sans ambiguïtés sur le statut des résultats invoqués. Au départ, ils sont tous considérés comme étant à démontrer, mais deviennent établis dans le déroulement de la rédaction.

Les différences entre les modes proposés concernent :

- le lien entre les raisonnements élémentaires
- le moment où on considère qu'un résultat est établi.

Mode 1 :
- tous les fragments intervenant dans la preuve doivent être cités;
- on peut enchâsser deux raisonnements uniquement dans la configuration suivante : on finit la démonstration de R1 et R2 se déduit directement de R1 sans apport d'hypothèse, d'outil ou d'autre résultat.

Exemple : H donc R1 ou R2
H donc R1 donc R2
R1 car H donc R2 (le «ou» est refusé ici)

Mode 2 :
- tous les fragments intervenant dans la preuve doivent être cités;
- on dispose du «or» pour introduire directement la preuve d'un nouveau résultat qui dépend du résultat que l'on vient juste de montrer; l'emploi de «or» n'est accepté ici que si le dernier fragment est le résultat qui vient juste d'être démontré.

Mode 3 :
- tous les fragments intervenant dans la preuve doivent être cités;

- on peut utiliser «or» et «donc» sans restriction, du moment que le résultat qui vient juste d'être démontré permet l'obtention d'un nouveau résultat (i. e. ne nécessite pas de démontrer d'autres résultats non encore obtenus).

Mode 4 :
- les résultats déjà établis peuvent ne pas être cités à nouveau. Ainsi, par exemple, si NR se démontre avec R1 et R2, deux résultats qui ont été prouvés au cours de la démonstration, on pourra écrire indifféremment :
- R1 et R2 donc NR
- R1 donc NR
- R2 donc NR
- NR

  La citation de NR indique la fin de la preuve de ce résultat. On ne pourra pas écrire NR car R1, les résultats à citer devant impérativement précéder ce qu'il faut montrer.

IV.1.d.2.b Aides automatiques

Pour des raisons de simplicité dans la création, les messages explicitement déclarés sur les erreurs de liens entre les fragments ont été supprimés. En contrepartie, différents types d'aide sont fournis automatiquement sur ce genre d'erreurs, précisant certaines relations entre les fragments proposés. On peut choisir entre trois niveaux qui permettent de filtrer l'apparition de ces messages qui donnent des indications très précises :

- le niveau 1 : uniquement sur un résultat qui est une conséquence directe d'un autre,
- le niveau 2 : lié au fragment choisi (à quel autre fragment est-il lié).
- le niveau 3 : lié à ce qu'il faut faire (en fonction de ce qui a déjà été fait).

Il y a quatre types de message :

- Il faut d'abord prouver <frag>.
- Il faudrait prouver par exemple <frag>.
- Ceci permet plutôt de prouver <frag>.
- Ce résultat se démontre en utilisant par exemple <frag>.

Les autres messages généraux sont évidemment toujours disponibles. La possibilité de revenir en arrière sur un choix de fragment, permet de comparer ces divers types d'aide (on peut changer le niveau d'aide à tout moment, il peut être fixé par défaut par l'enseignant, tout comme le style de vérification).

IV.1.d.2.c Impression / sauvegarde

On peut sauvegarder et imprimer toutes les démonstrations faites, et on a la possibilité d'associer un style particulier (normal, gras, condensé) à chacun des types de fragment : hypothèse, outil, résultat à démontrer, résultat déjà démontré, conclusion. La distinction entre résultat à démontrer et résultat déjà démontré permet de visualiser un aspect dynamique de la démonstration en imprimant différemment un même fragment suivant son statut dans le cours de la démonstration (voir annexe D.b, page 242).

IV.1.e Quelques problèmes généraux

Certaines insuffisances d'ARRIA n'ont pu être réellement palliées avec Cre-Arria. On peut mentionner rapidement :

- manque de prise en compte du côté stratégique. Même si ce n'était pas l'idée originale d'ARRIA qui était vraiment centrée sur la rédaction, la phase de typage des fragments est insuffisante, les élèves devant faire un travail annexe sur papier afin de préparer leur démonstration;
- manque de flexibilité dans la rédaction :
  - des vérifications avec d'autres styles de rédaction,
  - outils implicites intégrables,
  - pas de sortie de l'univers clos,
  - manque de capacités téléologiques;
- des types de raisonnement ne sont pas implantés (contraposée, récurrence, etc.);
- minimisation du rôle de la figure;
- problèmes de nature technique : écrans de visualisation trop petits, manque d'intégration des ressources dans les micro-ordinateurs de type PC.

On peut regretter l'absence d'un résolveur automatique, qu'il semble encore impossible de construire à l'heure actuelle, face à la somme de connaissances et de méthodes à intégrer. Plus particulièrement, la relation de Chasles est très difficile à maîtriser dans des systèmes de résolution, son utilisation amenant des risques d'explosion combinatoire. En fait, ARRIA est un mélange de bêtise et d'intelligence : on ne dira jamais assez que, dans la conception des outils d'enseignement, le but est que l'intelligence soit chez l'utilisateur pas dans la machine! Suivant les objectifs assignés, les contraintes d'un environnement têtu sont peut-être plus productives.

Chapitre IV.2

Évaluation d'ARRIA

Les premières évaluations d'ARRIA, dans le contexte scolaire et auprès de divers formateurs amènent des réflexions qui dépassent largement le cadre de cette réalisation. Cette partie tente de dresser un panorama assez complet des interrogations suscitées par les différentes réactions observées. En particulier, je remercie M. Jacques Arsac d'avoir accepté de me livrer ses cogitations, après avoir assisté à une séance avec ARRIA dans une classe de seconde.

IV.2.a ARRIA et les utilisateurs

Il n'y a pas eu d'évaluation formelle d'ARRIA, mais il a pu être utilisé et testé dans différentes classes. Ceci confère une subjectivité certaine aux expérimentations, d'autant plus que cet outil doit s'intégrer à une démarche, et qu'il est difficile de savoir si les succès ou les échecs sont imputables au logiciel ou à la démarche suivie.

J. B. Melet a testé ARRIA dans une classe de seconde de 33 élèves au cours de l'année scolaire 1988-1989. Voici ses conclusions :

« Chez les élèves qui ont des difficultés, j'ai trouvé un certain enthousiasme :

1. *La prise en compte de leurs difficultés, à leur rythme, sans que le professeur soit là pour les sanctionner, sur des règles qui sont simples (en particulier la distinction entre la cause et la conséquence) et surtout le fait qu'avec ARRIA, ils arrivaient à faire des démonstrations exactes.*
2. *Les meilleurs élèves ont approfondi leurs connaissances. L'un d'entre eux refaisait plusieurs fois la même démonstration, tout en changeant la rédaction, et il chronométrait le temps passé en essayant d'améliorer sa 'performance' en temps...*
3. *Tous les élèves ont apprécié les messages donnés par l'ordinateur et le fait que leur professeur soit disponible pour répondre à des questions plus délicates.*
4. *Tous les élèves sont actifs pendant l'utilisation du logiciel.*
5. *La méthode introduite par ARRIA a servi de référence toute l'année pour les autres démonstrations. »*

Ces réflexions amènent plusieurs commentaires :

1. Les élèves faibles réussissent à faire une production ce qui a nécessairement un effet bénéfique (voir image de soi), et prouve que l'aspect assistant d'ARRIA, aidant à produire une démonstration, est effectivement opérationnel. Ce phénomène, très courant d'ailleurs chez les élèves de l'école élémentaire, est trop souvent négligé par les adultes (qui désirent que le logiciel s'adapte à eux-mêmes alors qu'ils ne rencontrent aucune difficulté dans les activités proposées). A partir de ces premiers pas, il est plus facile de faire réfléchir les élèves sur leurs propres productions (aspect réification).
2. ARRIA est aisément détourné par des élèves qui maîtrisent suffisamment les activités proposées. Cette déviation, qui est capitale pour un usage véritable des outils informatiques, est difficilement prévisible, mais est tout à fait rassurante.
3. Ce point souligne le rôle d'un tel logiciel, intégré dans une séance de travaux pratiques. Les aspects d'incomplétude (voir IV.2.d) sont compensés par un mode cohérent d'utilisation.
4. L'activité semble réelle. Elle est facilitée par des objectifs clairement exprimés, et la simplicité de mise en oeuvre du logiciel.
5. Un travail avec ARRIA n'a de sens que s'il s'inscrit dans le travail général effectué dans la classe.

D'autres expérimentations ont été menées durant l'année scolaire 1989-1990. Malheureusement, il est difficile de se procurer des rapports, les auteurs étant rarement avertis ou consultés, sauf pour fournir une version du logiciel. Néanmoins, un formateur a pu donner une copie de son compte rendu pour le ministère. Il regrette certains défauts dans l'ergonomie du logiciel, et l'absence d'un module de création (imperfections qui sont réparées avec Cré-Arria) :
« Les élèves ayant des difficultés de raisonnement ont apprécié l'obligation qui leur était faite de tout justifier, ce qui peut les amener à corriger leur défaut d'imprécision... L'utilisation du logiciel dans sa version actuelle ne semble devoir représenter qu'une étape dans l'apprentissage des élèves. »
Il mentionne d'ailleurs différentes propositions d'amélioration dont certaines ont pu être prises en compte.

Dans les présentations réalisées par les formateurs académiques pour des professeurs de mathématiques, il ressort un phénomène intéressant. Le public se partage souvent en deux classes disjointes :

- ceux qui sont intéressés par l'approche d'ARRIA et souhaitent l'utiliser,
- ceux qui estiment que cette démarche ne correspond pas à ce qu'ils font et que le logiciel leur est donc d'aucune utilité.

Cette dichotomie montre que le choix de l'utilisation d'ARRIA est avant tout d'ordre pédagogique et que les enseignants sont tout à fait aptes à juger si un tel outil leur convient ou non, ce dernier n'étant pas adaptable à toutes les situations.

IV.2.b Rigueur et formalisme

L'une des premières objections que l'on peut faire vis-à-vis d'ARRIA est de regretter son formalisme. En effet, il s'agit plus d'un travail sur la logique que sur la géométrie. Or, la

logique, qui se préoccupe de représenter les énoncés et de vérifier leur validité ou non de manière indépendante de leur sens, est par essence formelle. « *L'adéquation à la réalité comme critère de vérité a été abandonnée; la vérité est celle des objets qui satisfont aux axiomes et, partant à toutes les propriétés qui en dérivent.* » [LOI 82] Bien qu'il puisse y avoir des raisonnements valides sans recours à la logique [JOHNSON-LAIRD 83], une certaine familiarisation avec la structure des raisonnements semble indispensable [CUPPENS 88].

Cette objection se complique du fait qu'ARRIA est un compromis entre la logique formelle et le langage naturel. La formalisation se traduit par des contraintes qui sont en partie purement conventionnelles. C'est une tentative de donner un statut clair aux rédactions faites par les élèves, de supprimer le flou et la subjectivité de leur correction.
« *'Parler avec rigueur de ce qui est approximatif', la formule semble paradoxale. C'est en effet une sorte de défi présenté à l'activité intelligente de l'homme :*

- *d'une part l'exigence de certitude et de rigueur;*
- *d'autre part l'inaccessibilité de cette perfection.* » [GUILBAUD 85] (citation un peu détournée puisqu'elle vise plus les problèmes d'approximation dans les calculs)

Le refus de toute formalisation n'est finalement pas très éloigné d'un refus d'exigence de rigueur. Tenter de s'éloigner des abstractions vides de sens est certainement une bonne chose, mais cela ne doit pas s'effectuer au détriment de la rigueur. Opposer d'ailleurs cette dernière à la créativité n'est pas un argument très convaincant : « *Ce souci contemporain de rendre les mathématiques plus rigoureuses n'a pas été un obstacle à leur développement. Bien au contraire, le formalisme a été la principale source de progrès des mathématiques contemporaines et doit nous éclairer sur leur véritable nature.* » [LOI 82]

A la limite, on débouche sur une notion d'«art» attaché à la démonstration et donc résistant à toute espèce de formalisation (la bosse des maths?).

De nombreuses critiques ont été faites sur l'emploi de certains connecteurs (le «or», le «ou» comme indicateur de réécriture), mais il n'y a jamais aucune proposition constructive sur les choix dans la grammaire utilisée (ses contraintes sont souvent mal comprises). De même, on reproche (voir par exemple [GUIN 90]) à Geometry Tutor [ANDERSON & Al. 85] une certaine lourdeur, puisque l'élève est obligé de tout déclarer, mais il est difficile de préciser quelles sont les tolérances acceptables. Les choix d'ARRIA sont tous contestables, encore faut-il que les objections ne cachent pas un refus de prise de position.

IV.2.c Outil générique vs scénario pédagogique

D'autres types d'objection proviennent du scénario même suivi par ARRIA. Il faut noter qu'il y a d'ailleurs une forme de paradoxe, puisque ce scénario n'est pas une création isolée d'un informaticien peu au fait des problèmes d'enseignement, mais qu'il a été proposé par des professeurs de mathématiques pour tenter d'apporter une solution à une difficulté réelle des élèves et qu'il a été labellisé par un jury placé sous l'égide du ministère de l'Education Nationale. Les écarts entre ARRIA et ce scénario sont minimes, son esprit ayant été scrupuleusement respecté. Les raisons de ce revirement sont complexes à cerner, l'une des hypothèses les plus plausibles est que la nouveauté des outils informatiques ne permet pas encore de percevoir concrètement une réalisation à partir de sa description 'papier' (notion de fantasme, voir IV.2.d, page 132).

On peut distinguer quatre points d'achoppement : l'idée de puzzle, la distinction entre la recherche et la rédaction, l'insertion dans un cursus, la préférence actuelle d'outils ouverts.

IV.2.c.1 Le puzzle

Les élèves ne sont pas forcément à l'aise face au logiciel, qui les oblige à respecter des contraintes inhabituelles. Tout d'abord, en ce qui concerne le choix des fragments, le logiciel leur impose de reconstituer un cheminement qui n'est pas obligatoirement celui qu'ils désirent suivre. En fait, quelqu'un a mis la pagaille dans une démonstration et il faut reprendre un raisonnement étranger et le remettre en ordre. Cette règle du jeu est parfois difficile à accepter pour les adultes qui revendiquent une liberté totale et acceptent mal d'être contraints par une machine. Les élèves n'ont pas ce genre de préoccupation et ont même rarement le loisir de formuler de telles exigences (l'enseignement les a depuis longtemps habitués à respecter les volontés, même étranges, du professeur). Ensuite, le respect des règles d'usage des connecteurs ne semble pas poser de réelles difficultés. Ils sont prêts à adhérer à des règles si elles s'avèrent productives (d'où l'intérêt de pouvoir mener au bout une démonstration à l'aide du logiciel). Ces contraintes, si elles sont correctement explicitées avec leurs limites, ont par contre l'avantage de ne pas faire croire à l'utilisateur que la machine comprend réellement le raisonnement suivi, mais qu'elle se contente de vérifier si l'enchaînement des propositions les respecte. Il n'y a aucune simulation d'accès au sens, même si certains messages sont très contextuels.

IV.2.c.2 La dichotomie forme/fond

ARRIA repose sur la distinction entre la recherche et la rédaction, que l'on peut voir comme une opposition forme/fond ou sémantique/syntaxe. Cette dualité recelle peut-être certains dangers. Ce point a déjà été discuté (IV.1.a, page 113), mais il faut rappeler que traiter la forme spécifiquement, de manière indépendante de la recherche, c'est comme maîtriser les opérations pour savoir comment les choisir (VI.3, page 193) : c'est nécessaire mais non suffisant. Des activités complémentaires sont sans aucun doute utiles (elles sont faites de toutes les façons) pour mieux comprendre les relations entre la découverte d'une preuve et son exposition. Ce n'est pas en 'trouvant' que l'on sait rédiger, comme un effet de bord automatique ou une génération spontanée. La rédaction est avant tout un acte social dont il faut posséder le code. Il y a une certaine forme d'imposture à prétendre que la forme est secondaire et à l'interpréter comme la trace du raisonnement, pour en déduire que le raisonnement est mauvais.

IV.2.c.3 L'utilisation pédagogique

ARRIA est indissociable du cadre dans lequel il est employé et de la mise en scène créée par l'enseignant. Ceci est en fait valable pour tout logiciel scolaire (et même de formation), ce qui a pu conduire à la position extrême de refuser toute analyse a priori des logiciels scolaires, même les pires pouvant être correctement utilisés, leurs limitations ou insuffisances engendrant des situations productives. Je ne souscris pas à cette affirmation qui mésestime les possibilités importantes des ordinateurs, en notant cependant qu'une bonne partie de

l'efficacité des logiciels repose encore sur l'enseignant. Dans le cas présent, on est peut-être amené à douter de la faculté des enseignants à intégrer ARRIA valablement dans une démarche intéressante, ce qui peut conduire à des déviations dangereuses. C'est un nouveau paradoxe puisque l'on oscille entre une croyance aveugle dans les capacités illimitées des enseignants (voir complétude, IV.2.d, page 132) et une méfiance profonde dans les outils qu'ils choisissent. Le fait que les professeurs se déterminent de manière claire dans le choix ou le refus d'ARRIA (IV.2.a, page 127), suivant qu'il s'adapte ou non à leur vision pédagogique est plutôt un signe encourageant montrant qu'ils perçoivent judicieusement les capacités de cet outil.

En fait, il peut s'agir d'un problème plus général lié au fait que les techniques utilisées et les connaissances représentées dans ARRIA ne sont pas celles des enseignants. Elles n'ont pas de caractère institutionnel ce qui rend leur usage suspect. Il n'y a pas de consensus sur les contraintes de rédaction, et ces dernières correspondent à des positions individuelles censées être contrôlées par la compréhension qu'ont les enseignants des raisonnements effectués par les enfants. Les réductions imposées par ARRIA et, plus particulièrement, la pauvreté des rédactions possibles peuvent être jugées trop contraignantes. Cependant, le travail sur la démonstration ne peut en aucun cas se limiter à des séances avec le logiciel. Il ne s'agit pas d'imposer cette forme de rédaction, qui ne constitue qu'un éclairage ou une étape. Améliorer, transformer, réécrire les démonstrations produites avec ARRIA est une activité importante : les rédactions d'ARRIA ne sont pas des modèles types à reproduire!

Une dernière remarque concerne les possibilités de collaboration offertes par ARRIA. Les multiples rédactions permises (même si elles ne diffèrent que par des aspects de surface) renforcent l'aspect social de la démonstration et encouragent les comparaisons (voir collaboration).

IV.2.c.4 La préférence pour les outils ouverts

Alors que les premiers logiciels d'EAO diffusés ont déçu les utilisateurs du milieu scolaire, de nombreuses personnes prônent l'usage d'outils ouverts, en excluant toute forme de contrainte pédagogique supposée restreindre la liberté de l'enseignant. Cette position est renforcée par un courant très fort encourageant le développement des mathématiques dites «vivantes» dans l'enseignement obligatoire, favorisant la créativité des élèves. La réalité semble cependant moins idyllique, les outils n'étant souvent pas introduits dans le cadre d'activités exploratoires mais plutôt détournés de cet usage par les enseignants dans le cadre de séquences pédagogiques ou de séances de travaux pratiques ne laissant absolument aucune initiative aux élèves. Il s'agit plus pour les enseignants de se réapproprier les outils en les considérant finalement comme des langages auteur spécialisés leur permettant de bâtir leurs séquences (ces remarques réfèrent à certains usages des logiciels EUCLIDE et CABRI). De plus, ce type de comportement n'est pas celui du professeur 'standard', qui a peu de temps à consacrer à des adaptations importantes des logiciels et à la création de ce type de séquences, mais concerne plutôt des enseignants plus avancés qui diffusent ensuite leur travail. En conséquence, pour l'utilisateur final de ce type de logiciel, la différence entre un outil ouvert à l'origine et un logiciel plus fermé est somme toute faible, et semble largement un épiphénomène.

Stratégiquement, il semble préférable de fournir des environnements les plus modifiables possibles (Cré-Arria est ainsi une évolution importante), tout en sachant cependant que ces ouvertures seront finalement peu utilisées. C'est en tout cas moins risqué que de fournir l'accès

à des connaissances mathématiques où certaines erreurs sont difficilement évitables et pour lesquelles les testeurs sont peu enclins à la clémence! (voir LYRE, III.2.b.3, page 95).

Remarque : dans le cadre de l'Ecole Normale, malgré une entente des divers intervenants, il n'a pas été possible d'organiser un échange des productions complétant des logiciels d'usage assez répandus (comme ELMO par exemple). Ceci souligne bien le fait que l'ouverture est plus une revendication théorique qu'une nécessité pratique.

IV.2.d Problèmes de complétude

Une dernière objection, d'une nature plus fondamentale, est celle de la complétude. Doit-on exiger la perfection des programmes utilisés dans un cadre de formation? Peut-on accepter qu'une machine puisse refuser une proposition faite par un élève alors qu'elle peut être considérée comme correcte? Ainsi, un des reproches majeurs à l'encontre de Geometry Tutor est son refus des pistes non reconnues. A ce titre, les outils observent une certaine neutralité, la difficulté intervient face à un environnement prescriptif.

A priori, on peut distinguer deux points :

- capacité de la machine à résoudre un problème donné,
- capacité de la machine à 'comprendre' pour accepter ou refuser la proposition d'un élève.

En fait, le deuxième point se ramène au premier puisqu'en l'état actuel, la 'compréhension' de la piste suivie par un élève ne peut se faire que par comparaison avec les pistes que l'ordinateur peut générer et qu'il n'est pas possible à la machine de faire intervenir des connaissances non contenues dans la situation courante. Ceci est dû à l'impossibilité (au moins actuelle) d'accéder au sens.

IV.2.d.1 Complétude des résolveurs

On peut sans conteste affirmer que les programmes se limitant aux opérations numériques sont complets (i. e. capables d'effectuer les calculs arithmétiques). On peut simplement rencontrer des problèmes de précision, mais les machines surpassent largement les capacités humaines. En fait, dans les domaines parfaitement algorithmisés, si le problème courant est dans la bonne classe, il ne doit pas y avoir de difficulté.

En calcul formel, si on impose aux machines de respecter une certaine plausibilité psychologique (II.2.a.1, page 49), on arrive plutôt à une notion de quasi-complétude (liée à la complexité d'un problème, VI.3.c) : lorsqu'une méthode connue s'applique à un problème, le logiciel doit être capable de l'appliquer (c'est une forme de complétude par les méthodes). On admettra que certains problèmes puissent ne pas être résolus, soit parce qu'ils nécessitent des choix très particuliers, soit qu'ils sortent du cadre proposé (nécessitant des méthodes non connues du logiciel). Dans un cadre non formel, les obstacles interviennent plus sur les connaissances générales, du fait que l'ordinateur ne maîtrise pas des choses que l'on peut juger comme évidentes (pour des humains).

ARRIA se réfère en définitive aux idées classiques en intelligence artificielle pas tant dans les générations de chemins à partir des liens entre les fragments que dans sa définition restrictive de la grammaire. ARRIA cumule finalement deux types d'incomplétude :

- celle liée aux choix de fragments et aux relations entre eux,
- celle liée à la grammaire.

On peut essayer de réduire la première incomplétude, en recensant le maximum de pistes possibles (et en effectuant des mises à jour). En création, c'est l'enseignant qui assume cette responsabilité, les outils de vérification qui lui sont fournis sont encore rudimentaires (simple vérification de l'existence d'un chemin), ils pourraient être améliorés. La deuxième est plus structurelle et fait partie intégrante du scénario : c'est l'absence de négociation possible avec la machine qui contraint l'élève à respecter les règles édictées.

IV.2.d.2 Usage des systèmes experts

Dans d'autres domaines que l'enseignement, ce problème de complétude ne semble pas aussi crucial. Ainsi des systèmes experts sont utilisés quotidiennement. L'organisation globale tient compte de leurs limitations. En fait, on les intègre le plus souvent comme des outils d'aide à la décision ou d'aide au diagnostic. Ils fournissent une expertise qui est la plus fiable possible mais qui n'est en aucun cas garantie. Les études sur MYCIN [YU & Al. 79] ont montré que la machine donnait des résultats d'une fiabilité comparable à celle des experts du domaine. Il s'agit cependant bien d'une aide, la responsabilité (voir II.1.c.4.c, page 46) étant assurée par une personne.

Dans le cadre de l'enseignement, il faut d'abord remarquer que les connaissances mises en jeu (au niveau des domaines enseignés) sont plutôt moins expertes (elles sont répertoriées dans les programmes scolaires). Par fonction, l'enseignant est supposé les connaître parfaitement. Cependant, ce sont des savoirs transposés qui sont souvent erronés (la réduction dite pédagogique est acceptable dans les programmes scolaires, pas dans les programmes informatiques!). Ensuite, les outils, quels qu'ils soient, sont encore fortement étrangers au monde scolaire : on ne reconnaît pas encore la nécessité de disposer d'autres sources d'informations que la parole de l'enseignant. Il y a une part de méfiance instinctive de la technologie, mais aussi le fantasme diffus de la machine prenant la place de l'enseignant. (Voir le titre paru dans la presse locale pour présenter l'université d'été européenne qui s'est tenue au Mans sur les tuteurs intelligents : « *Remplacer la maîtresse d'école* »)

IV.2.d.3 Complétude des enseignants

Il peut s'avérer intéressant de faire une comparaison entre les enseignants, les livres et les logiciels :

- l'enseignant est incomplet (en pratique), mais cette incomplétude est non vérifiable, non reproductible et non institutionnelle (complétude théorique liée à la fonction exercée),
- le livre est incomplet, mais ses erreurs sont sensées être corrigées par le professeur. Elles sont reproductibles (elles sont écrites, Erratum), mais non institutionnelles : ce sont les éditeurs qui font les livres, les enseignants ont la liberté du choix.
- le logiciel est incomplet (ou alors il traite de domaines très restreints et son adaptabilité à l'élève est quasi nulle). Des bogues sont de plus inévitables (comme dans les livres), mais ils n'apparaissent pas toujours de manière évidente. Ils sont reproductibles et

vérifiables. Mais, curieusement, si on reconnaît aux enseignants les capacités de dominer les livres, on les considère démunis devant les logiciels.

Dans le cas d'un hypertexte, sorte de logiciel/livre, les problèmes de complétude se posent aussi d'une manière duale, dans l'apparence d'une fausse liberté (voir III.3.b, page 107).

Cette problématique de complétude apparaît comme une difficulté à définir un statut clair d'utilisation des ordinateurs dans un cadre de formation. On constate une oscillation constante entre des phases de sur-estimation et de sous-estimation de leur pouvoir et de leur rôle. Un état stationnaire est actuellement trouvé dans la notion générique d'outil, qui permet d'évacuer toute référence explicite à des techniques pédagogiques. L'outil peut se valider hors d'un contexte réel de formation et se justifie par ce qu'on (ce «on» étant assez mal défini) est capable de réaliser avec, non par les difficultés d'apprentissage qu'il peut aider à résoudre (d'une certaine manière la forme est prédominante par rapport à l'objectif). Cette vision ne prend pas en compte les réalités pratiques du travail scolaire : elle consiste plus à éviter de poser le problème qu'à tenter de le résoudre.

Chapitre IV.3

Hyper-ARRIA

HYPER-ARRIA est un projet d'extension du logiciel ARRIA pour faciliter des tâches nécessitant la structuration d'un raisonnement. Ce système n'est pas conçu dans une perspective de réalisations de tuteurs d'auto-formation, mais comme générateur d'environnements informatiques utilisables en complément d'une formation traditionnelle. Dans ce cadre il intègre un module AUTEUR destiné au formateur en situation pour lui permettre d'ajouter rapidement ses propres exercices. Une aide à l'évaluation globale du travail de l'apprenant est prévue, la gestion générale de l'apprentissage restant l'apanage du formateur.

IV.3.a Au delà de ARRIA

IV.3.a.1 Description formelle d'ARRIA

L'architecture d'ARRIA telle qu'elle a été décrite (IV.1.c, page 117) est générale et ne dépend pas d'un domaine d'étude particulier. On dispose :

- d'un ensemble fini de fragments F_i ($1 < i < n$), qui peuvent être spécifiques à une situation ou génériques (ce qui correspond aux outils).
- de propriétés associées à chacun de ces fragments, ces propriétés prenant leurs valeurs dans des ensembles finis ou ordonnés. On a des listes de couples $< attribut, valeur >$: $(P_1,V_1),\ldots (P_n,V_n)$, qui sont partiellement ou complètement indépendantes du contexte (valeurs intrinsèques gardées par le fragment quel que soit le contexte où on l'utilise);
- des relations entre les fragments telles que : prouve, précède, similaire, oppose, etc.
- des liaisons : connecteurs, signes de ponctuation, opérateurs, ...

A partir de ces éléments on définit une grammaire liée au domaine considéré. On construit un analyseur pas à pas (ou global) qui s'appuie sur les listes attribut-valeur et les liaisons (syntaxe) et sur les relations définies entre les fragments (sémantique et pragmatique). Rappelons qu'il ne s'agit pas uniquement de contrôler la correction des rédactions proposées, mais d'associer des messages adaptés aux diverses situations susceptibles d'être rencontrées (par exemple des associations erronées de fragments).

Les diverses propriétés et relations ainsi que les fragments peuvent être :

- entièrement déclarés par un auteur,
- partiellement déclarés (valeurs automatiques, fragments prédéfinis, etc.),
- générés par le système.

Au niveau utilisateur, les interactions sont générales :

- lecture active d'un énoncé (recherche d'information, III.1.b, page 85)
- classification ou typage (valeurs associées aux propriétés),
- rédaction en pas à pas contrôlée,
- impression et sauvegarde éventuelles.

IV.3.a.2 Problématique d'extension

Les réflexions autour d'ARRIA et du générateur CRE-ARRIA montrent l'intérêt d'étendre ce travail, à la fois dans la phase de rédaction, et dans une phase à définir de planification (spécification générale du cheminement de la démonstration). Une idée qui semble productive consiste à faire coexister à la demande des multiples perspectives (à rapprocher de la notion de point de vue, II.2.b.5) :

- des présentations différentes adaptées à une classe d'utilisateurs, en utilisant la même spécification des exercices pour l'auteur;
- offrir un générateur d'hypertexte au formateur pour enrichir l'environnement de résolution et donner le maximum de pistes;
- faciliter le transfert des exercices créés pour rentabiliser le temps nécessaire à la conception d'un exercice significatif.

L'étape de planification préalable à la rédaction s'impose si on veut prendre en compte plus complètement l'activité de résolution. Elle peut consister, en dehors du typage des fragments, à la constitution de blocs et à la déclaration de liens entre ces blocs. Ce travail, comme la partie rédaction, peut s'effectuer dans une interaction permettant un traitement plus global s'éloignant d'un pas-à-pas qui peut être pénible. On peut s'inspirer, par exemple, de l'interface de BRIDGE ([BONAR & CUNNINGHAM 88]) où l'utilisateur place les divers fragments comme il veut (représentation plane) et interroge le système à la demande. On retrouve l'aspect plan et non linéaire de la démonstration et on se rapproche plus de l'idée de puzzle. Ceci suppose d'être capable de concevoir une aide en ligne «intelligente» et de reprendre le système de vérification pour qu'il puisse fonctionner d'une manière analogue à un compilateur. En particulier, il faut pouvoir préciser le mode d'arrêt sur les erreurs (ordre chronologique, "warnings", pause,...) et valider les parties éventuellement correctes.

IV.3.b Structure générale d'Hyper-Arria

On peut décomposer l'étude d'Hyper-Arria en trois parties, correspondant à trois points de vue :

- l'ensemble des ressources générales intégrant les outils logiciels indépendants du domaine,
- les exercices et la couche d'explication créés par l'auteur,
- le mode utilisateur.

IV.3.b.1 Mode utilisateur

Tout d'abord, l'utilisateur a accès à une documentation générale en hypertexte sur le domaine qu'il peut consulter comme il le désire et en extraire à loisir des parties pour les imprimer. Il peut lancer des démonstrations automatiques pour voir comment se déroule une session. Il peut enfin choisir un exercice avec l'aide du système. Le travail principal consiste bien sûr à résoudre un exercice. Cette résolution se déroule en trois étapes :

1. **Lecture active de l'énoncé**
   Travail sur l'énoncé en utilisant divers outils d'aide à sa lecture et à sa compréhension. Cette étape est largement exploratoire, des accès à l'hypertexte peuvent fournir tous les éléments nécessaires pour la découverte de la solution. Il s'agit bien de relier des connaissances générales à un contexte particulier (voir III.2.a, page 91).
2. **Tri et Planification**
   Un ensemble des fragments est proposé. Plusieurs travaux peuvent être alors effectués :
   - typage : indication de la nature de chacun des fragments;
   - constitution de blocs : mise en commun d'un ou plusieurs fragments;
   - déclaration d'un plan de résolution ou reconnaissance des étapes importantes, dans un mode plus ou moins guidé. On peut ainsi se contenter de sélectionner des résultats importants, réaliser un arbre à l'aide d'une souris (à partir des blocs ou de certains fragments). Cette étape est vérifiée ou non suivant le choix de l'utilisateur et/ou du formateur.
3. **Rédaction**
   Plusieurs modes sont possibles, reliés ou non à l'activité de planification précédente :
   - mode linéaire : introduction et vérification des fragments un à un;
   - mode plan : on ordonne certains fragments comme on le désire, en faisant toute ou une partie de la rédaction, avec aide et vérification à la demande;
   - mode 3D : on reprend le plan déclaré à l'étape précédente, on détaille localement chacune des parties et on articule les diverses parties entre elles.

   Si la rédaction a pour finalité la production d'un document, de nature informatique ou non (texte, ou texte + images), ce dernier peut ensuite être sauvegardé pour une utilisation externe (impression ou intégration à d'autres ressources).

L'utilisateur a toute latitude pour naviguer entre ces étapes. Il peut en sauter, y revenir, accéder à l'hypertexte à tout moment. Il peut successivement faire plusieurs rédactions. A l'issue d'un exercice (terminé ou abandonné), d'autres exercices sont proposés par le système, qui tient compte du cursus déjà suivi, des erreurs et réussites constatées précédemment.

IV.3.b.2 Ressources "système"

- un hypertexte général sur le domaine (qui reste constamment accessible) intégrant texte, images, animations ainsi que divers processus et simulations. C'est un livre électronique, consultable par l'utilisateur et «appelable» par le module de vérification.
- un ensemble d'interactions partiellement paramétrables sur chacune des phases. En particulier, un gestionnaire de fragments permettant une manipulation aisée. On utilise des éléments d'une boîte à outil classique : menus, fenêtres, souris, etc., mais aussi spécifique Hyper-Arria.
- un système de vérification paramétrable des types de rédaction admissibles, des types de classification et planification.
- un générateur de démonstrations automatiques.
- un gestionnaire de modèle de l'apprenant, de type relativement rudimentaire.
- un gestionnaire de cursus pour faciliter le choix des exercices à traiter.

IV.3.b.3 Déclarations à faire par le formateur

Il dispose d'un éditeur spécialisé intégrant des vérifications de cohérence (en particulier sur l'arbre de résolution). Pour créer un exercice, il doit déclarer :

- la liste des fragments avec les descripteurs associés,
- éventuellement la constitution des blocs,
- un arbre et/ou (lien entre les fragments ou les blocs),
- des relations complémentaires entre les fragments et/ou les blocs.

A ceci s'ajoute un hypertexte comprenant l'énoncé et diverses ressources associées (figures par exemple) ainsi que des messages spécifiques pouvant préciser des erreurs sémantiques liées au problème. Cet hypertexte est inclus automatiquement dans l'hypertexte général lié au domaine. Les paramétrages sur le type d'interaction souhaité ou le style de vérification peuvent être associés au problème considéré ou précisés dynamiquement en cours de session.

Le formateur peut choisir des descripteurs associés au problème qui seront pris en compte pour le choix des exercices et le suivi de l'apprenant. Il peut aussi réaliser des démonstrations automatiques.

Pour ce qui concerne la partie solution d'un exercice, l'idée est de sélectionner des descripteurs suffisants pour que le plus gros travail puisse être effectué de manière automatique par le système (sur l'arbre et/ou, ces descripteurs et les relations). En particulier, les vérifications générales correspondent pour une grande part à un système de règles sur les descripteurs. Notons encore que l'absence de résolveur n'est pas a priori dramatique, un tel module pouvant d'ailleurs être intégré dans le système. Il faut cependant remarquer que l'existence d'un résolveur automatique sur un domaine donné doit changer fortement l'apprentissage lié à ce domaine : ce dernier est-il encore vraiment utile? Ne faut-il pas plutôt former les personnes à l'utilisation cohérente de ce résolveur (II.3.b, page 73) ?

Remarque : ceci constitue une description abstraite des diverses composantes du système. Une réalisation ne contiendra pas obligatoirement la totalité des outils décrits. En particulier, le

manque de connaissances didactiques sur un domaine choisi peut rendre difficile l'introduction d'un module de suivi ou la gestion de divers profils. La qualité et la quantité des messages d'erreur prévus peuvent aussi fortement différer.

Le tableau ci-joint récapitule les caractéristiques générales d'HyperArria.

Tableau récapitulatif

	Ressources Système	Déclarations du formateur	Environnement utilisateur
Connaissances générales du domaine Messages	Hypertexte général Générateur (intégrateur) Catalogue d'erreurs standards	Compléments éventuels Ajout d'autres ressources (textes, images, processus simulations, démonstrations)	Consultation à la demande hiérarchique ou contextuelle Démonstrations automatiques
Exercices	Editeurs spécialisés Générateur (intégrateur)	Enoncé (texte, images, etc.) Fragments (blocs) avec les descipteurs et relations Arbre et/ou de résolution	Divers accès aux structures déclarées suivant les étapes
Vérifications des exercices	Module paramétrable Plusieurs grammaires	Paramétrage (type et mode) Choix d'une grammaire	Mode plus ou moins guidé
Messages d'erreur liés aux exercices	Hypertexte général Moule pour introduire les erreurs sémantiques	Ajout de messages hypertexte sur les erreurs d'association	Accès en situation à l'hypertexte
Interactions	Modes généraux fournis paramétrables	Choix des interactions lié : - type d'exercice, - profil utilisateur	Phase 1 : lecture active Phase 2 : typage, groupement planification linéaire, 2D Phase 3 : linéaire, 2D ou 3D
Gestion du profil et suivi	Modèle simple *"overlay"* Gestionnaire de cursus Plusieurs stratégies pédagogiques prévues	Descripteurs associés à un exercice Choix d'une stratégie pédagogique globale	Choix de l'exercice suivant (sur proposition) Choix du style d'interaction Accès au modèle

Certaines techniques complémentaires peuvent jouer un rôle déterminant pour la flexibilité de l'interaction :

- les déclaration et reconnaissance de plans (V.1.c.1, page 154), ce qui peut donner une étape intermédiaire entre l'énoncé et la rédaction et intervenir sur la liste des fragments à fournir. L'analyse a posteriori d'une rédaction complète ou partielle (i. e. n'intervenant pas durant sa construction) amène une interaction plus souple qui étend le simple 'modèle de trace' utilisé dans ARRIA. Un problème analogue est celui de PROUST [JOHNSON 88] ou de CHIRON [SACK 88] (V.1.c.6, page 158), ainsi que l'extension LISP-TUTOR [SKWARECKI 88]. Il faut noter que cela favorise aussi l'émergence de modèles mentaux.
- génération automatique de fragments pouvant être assurée par un démonstrateur dans le domaine considéré (IV.1.b.2, page 117).
- classification des graphes de démonstration et association d'interventions particulières.
- outils spécifiques de 'polissage' de la rédaction. En effet, si le logiciel traite avant tout la structure profonde de la rédaction, la structure de surface est aisément améliorable (ajouts d'autres connecteurs, écritures différentes des fragments, etc.).

IV.3.c Systèmes auteurs

IV.3.c.1 Généralités sur les systèmes auteurs

Les systèmes auteurs ont eu des résultats mitigés dans le cadre de l'EAO traditionnel (projet DIANE par exemple), à cause de la rigidité de leur architecture et l'insuffisance des rétroactions possibles. Voir une étude comparative [MADAULE & Al. 87] Dans le cadre d'ARRIA, différentes raisons montrent l'inadéquation de la démarche des langages auteur traditionnels :

- Le problème classique de l'analyse de réponse est étendu à un système de diagnostic en ligne. Ce dernier peut être du type 'model tracing' dans une interaction de type linéaire avec suivi pas à pas (comme dans ARRIA), ou être le résultat d'un processus dynamique de déduction. Une modélisation de l'élève est possible et peut diriger la recherche de diagnostic et commander les interventions (messages hypertexte ou choix des exercices).
- L'apprentissage est essentiellement basé sur l'activité de l'apprenant. Ce dernier résout effectivement des exercices non triviaux avec l'aide du système et peut à tout moment consulter les informations dont il a besoin.
- L'intégration d'un résolveur fonctionnant en ligne ou en différé (c'est à dire générant et stockant l'ensemble des solutions avant le déroulement d'une session de travail avec l'élève) est nécessaire pour contrôler efficacement le cheminement proposé.

Dans le cadre de l'EIAO, les recherches se poursuivent pour fournir des outils génériques destinés à la création de tuteurs intelligents. D'après Johnson [JOHNSON 88] le développement et l'utilisation de ce type d'outils devrait être une priorité pour les cinq années à venir.

L'intégration des techniques d'intelligence artificielle peut résoudre certaines limitations des systèmes auteurs traditionnels. Malheureusement, certains obstacles majeurs subsistent, le plus important étant de pouvoir atteindre un consensus sur les caractéristiques de tels langages [SELF 85a]. Les travaux s'orientent souvent vers un aspect organisationnel de la matière à apprendre en molécules articulées ensuite aux moyens de structures gérées par des interprètes généraux (gestion du curriculum, voir III.1.b.3). Voir par exemple [LAVOIE & Al. 89] où divers projets sont mentionnés : MEDIAN (Carton, Quéré), APT (Fine), ADAPT (Mudrick).

Dans certains systèmes, l'organisation générale du tuteur est entièrement prise en charge (par un 'expert pédagogue'), mais les activités de base de l'utilisateur doivent être programmées. C'est le cas du système GAMETE [REGOURD 88] qui utilise LISP ou LOGO. Dans le cadre du projet Héron (exemples de prototype avec ART [GAUTHIER & IMBEAU 89]), la principale composante de l'interface est de type micromonde. Une approche un peu duale est de partir d'un système expert existant et de l'adapter pour l'utiliser dans un cadre d'apprentissage (introduction de points d'arrêt [CLANCEY & JOERGER 88] par exemple).

On arrive à la conception de systèmes ouverts donnant des outils pour l'auteur mais aussi la possibilité d'utilisation d'un langage d'Intelligence Artificielle, et des ateliers de conception de didacticiels (ArbraCas [TIJUS & Al. 90], STRAGUIDE [CLAES 88], ORGUE [BESSIERE & Al. 89], etc.). Dans ce domaine, les techniques hypertextes et hypermedia semblent prendre une place prépondérante : elles offrent à l'utilisateur un environnement plus complet et plus attractif (III.1.b.3, page 88), essentiel dans la formation professionnelle. Des outils d'aide à

l'auteur se développent, comme le système IDE, Instructional Design Environment [PIROLLI & RUSSELL 88].

IV.3.c.2 Spécificités d'Hyper-Arria

L'approche d'Hyper-Arria se distingue des approches précédentes en privilégiant non pas une organisation par les contenus d'enseignement mais par les interactions. Elle n'est pas universelle, mais correspond à différents domaines, ce qui facilite les possibilités de transfert (II.3.a.4, page 71).

Tout d'abord, il faut noter un renversement de tendance par rapport aux langages auteur. Ces derniers privilégient une logique d'exposition, proche du paradigme des langages impératifs : on crée un déroulement, en répartissant les connaissances le long des divers chemins prévus pour les utilisateurs. On y oppose une logique hypertexte où l'utilisateur accède directement à la connaissance, et reprend partiellement le contrôle de son apprentissage. Les systèmes d'EIAO permettent eux plus de flexibilité en ajoutant la possibilité de gestion du cursus. La déclaration générale des connaissances, à l'aide de relations, confère une grande indépendance vis-à-vis des modes d'exposition ou de travail.

L'investissement nécessaire à la production du système HYPER-ARRIA amène à adopter une perspective industrielle, adapter la logique d'usage des progiciels aux systèmes de formation informatisés. HYPER-ARRIA ne pourra intégrer directement toutes les fonctionnalités souhaitées mais s'inscrit dans une évolution, en assurant une compatibilité ascendante entre les exercices créés par les formateurs et le système général se complexifiant peu à peu. Ceci s'avère possible du fait que divers environnements de présentation et de travail récupèrent la connaissance déclarative générale associée aux exercices.

Il faut remarquer l'importance pratique de la notion de semi-résolveur. Les domaines intéressants à traiter ne permettent pas forcément la création de résolveurs. Ils peuvent être soit trop complexes, soit encore mal connus pour automatiser entièrement le processus de résolution. Dans ce contexte, un moyen terme consiste à créer des semi-résolveurs, c'est-à-dire des programmes qui ne trouvent pas directement la solution ou les solutions à un problème, mais sont capables de générer et vérifier un grand nombre de solutions admissibles à partir d'une spécification générale. Ce sont en quelque sorte des résolveurs «assistés». On peut d'ailleurs imaginer l'association de résolveurs locaux et de connaissances sur la solution (ou un schéma de solution) d'un problème particulier.

Il faut noter que le fait de pouvoir construire un résolveur est indépendant de l'interaction générale et peut être réintroduit par la suite. De plus, ce modèle amène de fortes contraintes pour ce résolveur qui doit être capable de vérifier toutes les solutions admissibles, en temps réel, et de fournir un diagnostic cohérent et intéressant pour l'apprenant en cas d'erreur. Un résolveur trop peu souple serait d'une utilité plutôt anecdotique, puisque son rôle se limiterait à accepter des solutions correctes, sans permettre de suivre l'activité d'un apprenant.

Dans le domaine spécifique des problèmes de géométrie scolaires, les divers démonstrateurs répertoriés (IV.1.b.2, page 117) peuvent sans difficulté générer les informations utiles à ARRIA pour contrôler la rédaction d'une démonstration. On peut facilement imaginer un dispositif analogue à Micro-Search [SLEEMAN 87a] dans lequel l'ensemble des cheminements possibles (d'une longueur maximale fixée à l'avance) est constitué et codé dans un exercice pour ARRIA.

Ainsi, l'approche de Hyper-Arria est pragmatique et s'appuie :

- sur un semi résolveur ascendant intégré,
- sur un langage auteur spécialisé déclaratif.

IV.3.d Domaines d'application

Hyper-Arria ne s'applique pas à tous les apprentissages, mais est spécialisé dans des domaines faisant intervenir des raisonnements et une communication de ces raisonnements sous une forme plus ou moins contrainte. Le fil conducteur reliant les diverses applications envisageables correspond à leur découpage possible en deux phases :

- étape de planification ou recherche d'un raisonnement global lié à la résolution d'un problème,
- étape de transcription ou d'écriture de la solution en respectant les contraintes propres au domaine étudié.

L'aide à l'écriture de démonstrations mathématiques a été la première instance créatrice du système. Il est clair que ceci peut s'étendre à des domaines voisins :

- apprentissage de la logique élémentaire : les exercices créés par les formateurs peuvent spécialiser le système à des domaines intéressant les publics qu'ils ont en charge. Ceci évite des présentations trop formelles et peut induire des apprentissages annexes. Une maquette a ainsi été réalisée sur l'usage du 'modus ponens' et du 'modus tollens'.
- étude de l'argumentation : on s'éloigne de la logique classique pour se rapprocher de logiques naturelles.

  Le domaine de l'apprentissage des langages de programmation peut être traité en partie avec Hyper-Arria : la phase de planification correspond à une spécification d'un problème, la rédaction à son codage effectif dans le langage désiré. Il ne s'agit pas de passer par un pseudo-langage, mais de séparer l'analyse de l'écriture d'un programme. Le PASCAL est un bon exemple de langage qui peut être abordé à l'aide d'Hyper-Arria. Ce domaine est finalement l'un des plus prolixes dans les recherches sur les tuteurs intelligents. Au congrès de Montréal en 1988, 20% des communications portent sur ce thème. Un survol général peut être trouvé dans [Du BOULAY & SOTHCOTT 87].

  Les langages abordés sont principalement LISP, PASCAL et PROLOG :

  - PROLOG : APROPOS [LOOI 88], DIPLOMAT [BARRIL 89], APILOG (III.2.b.2, page 93), [BRUILLARD & PEREIRA 87], micromonde [NICHOL 88],
  - PASCAL : BRIDGE [BONAR & CUNNINGHAM 88], MENO-II [SOLOWAY & Al. 83], PROUST [JOHNSON 86], CHIRON [SACK 88],
  - LISP : SCENT-3 [McCALLA & GREER 88], LISP TUTOR [ANDERSON & REISER 85].

  On peut citer aussi ADA (SAIDA [GRANDBASTIEN 88]), et l'étude des concepts généraux de l'informatique.
- conception de rapports, voire de résumés de textes informatifs, ou plus généralement aide à la réalisation de documents. Ces derniers s'entendent comme étant des ensembles structurés de textes et de graphiques organisés pour une finalité clairement définie.

On peut étendre le système dans une direction un peu différente, comme aide à la conception et à la réalisation de tâches procédurales complexes.

Il faut noter que le fait de construire ces environnements à l'aide d'interactions génériques confère une forme externe stable facilitant le transfert des apprentissages dans les différents domaines abordés.

IV.3.e Problématique de transfert

L'organisation interne d'ARRIA constituée de fragments ayant des propriétés et de relations entre ces fragments est finalement très voisine de celle d'un hypertexte contrôlé par un gestionnaire de navigation : trouver un chemin est analogue à la construction d'une argumentation. Une comparaison avec l'utilisation de démonstrateurs en géométrie est aussi intéressante. Elle est résumée dans le tableau suivant :

Résolution de problèmes	Démonstration
Trouver un plan	Traduire le plan
Recherche	Rédaction
Intelligence créative Heuristiques / Prototypes	Transcription mécanique (si maîtrisée) Algorithmique
Particularisation	Généralisation
Expertise locale	Connaissance générale
liée au domaine (performance)	Transferts possibles autres domaines
Apports en mathématiques (analyse du domaine)	Aucun apport (logique?)

Le problème des heuristiques et de leur enseignement est central (VI.3.b, page 194).

La dualité mise en évidence tient au fait fondamental que la démonstration ne reflète pas le processus de résolution. Des rédactions sur machine peuvent faire tomber cette contrainte, comme les travaux sur DYNABOARD [KALTENBACH & FRASSON 89]. On retrouve l'aspect essentiel de la réification.

Cinquième Partie

BADAUD
Base d'Aide au Diagnostic pour l'Assistance à un Utilisateur Débutant

" All told, the research brings Good News and Bad News. The Good News is that, basically, students are acting like creative young scientists, interpreting their lessons through their own generalizations. The Bad News is that their method of generalizations are often faulty. "

[MAURER 87]

" One thing we have learned is that there is no point in striving laboriously to diagnose and remediate huge libraries of superficial errors, any more than there is in destroying the leaves and stem of a dandelion weed; in order to eradicate the problem, an attack must be made on its roots. In the case of arithmetic, further bugs might otherwise arise as a result of an uncorrected basic misconception. "

[HENNESSY 90]

Chapitre V.1

Diagnostic et modélisation d'un apprenant

V.1.a Erreur et diagnostic

V.1.a.1 Exemples généraux

Le concept de diagnostic est, en première approche, une forme de généralisation de l'analyse de réponses chère à l'E. A. O. classique. L'idée est de trouver pourquoi un apprenant (ou plus généralement un utilisateur) ne fournit pas la réponse attendue à une question posée ou lance une requête incorrecte ou simplement ambiguë. Voici quelques exemples :

1) Quelle est la capitale de l'Australie?
→ Sidney

On ne peut considérer cette réponse comme totalement fausse : l'ambiguïté du mot «capitale» est peut être en cause. Ainsi, il ne s'agit pas forcément d'une lacune dans les connaissances géographiques de l'apprenant, ce que l'on pourra vérifier en reposant la question en indiquant qu'il s'agit de la capitale administrative et non de la ville la plus peuplée.

2) Calculer 2/3 + 3/4
→ 1/2

Dans une tâche procédurale, il faut reconstituer le cheminement ayant abouti à la solution fournie. L'erreur peut provenir soit :

- d'un manque de connaissances, la résolution faite par l'élève dépend des analogies qu'il peut trouver avec son propre savoir;
- de connaissances fausses correctement utilisées, comme l'application d'une méthode erronée;
- de connaissances correctes mais mal utilisées (erreur de calcul, erreur d'énoncé, etc.).

L'hypothèse la plus probable est que l'élève a effectué un produit en croix, enlevé les deux 3 puis simplifié la fraction obtenue (2/4). Ceci peut être dû au fait qu'il a confondu l'opérateur

'+' avec 'x', qu'il a pu être attiré par la proximité des deux 3 présents (erreur liée aux données de l'énoncé) ou qu'il emploie une mauvaise méthode d'addition de fractions.

3) Requête MS-DOS :
copy toto.exe
(Le fichier ne peut pas être copié sur lui-même)

Cette requête n'est pas directement exécutable, il est probable que l'utilisateur a omis de préciser la destination de la copie (à moins qu'il veuille faire une copie de sauvegarde de son fichier sur la même unité). Résoudre l'ambiguïté de cette requête consiste à «deviner» l'intention de l'utilisateur.

4) Requête MS-DOS :
copy toto.exe a:
(TOTO.EXE Fichier non trouvé).

Dans ce cas, il peut s'agir soit d'une erreur orthographique sur le nom du fichier, soit d'une erreur dans le nom de l'extension, soit une erreur de répertoire. Le message proposé par la machine n'est pas suffisamment clair pour aider un utilisateur débutant à proposer une requête correcte.

Le diagnostic, dans un système ayant une utilité pratique, doit permettre à la machine de dépasser le simple message d'erreur, afin d'avoir la meilleure intervention possible dans le contexte courant. L'E. A. O. classique résout ce problème de manière plutôt locale; l'introduction des techniques d'I. A. amène une vision plus globale liée à la construction d'un modèle de l'utilisateur, prenant en compte à la fois un aspect historique et le mode d'utilisation des connaissances dans la réalisation d'une tâche : on cherche à expliquer un comportement, qu'il conduise à une production correcte ou erronée.

V.1.a.2 Erreur et enseignement des mathématiques

La prise en compte de l'erreur dans l'enseignement des mathématiques est un phénomène récent. Ainsi, pour la plupart des enseignants, l'erreur « *... est interprétée comme l'indice que l'élève ne sait pas faire, n'a pas travaillé... et non pas comme l'indice que l'élève sait quelque chose d'incorrect, de partiel ou qui ne correspond pas à l'attente de l'enseignant, et donc qu'il faudra élucider avec lui, travailler pour, à partir de là, construire une connaissance correcte... L'erreur n'est pas considérée comme support possible d'un apprentissage... Les erreurs sont considérées comme parasites (comme des fautes?)* ». [CHARNAY 86]

Au contraire, les chercheurs se penchent de plus en plus sur les erreurs, en les considérant comme fortement liées à l'apprentissage. « *L'erreur est constitutive du sens même de la connaissance* » ([BROUSSEAU 83] Les erreurs proviennent d'obstacles épistémologiques (contenu du savoir [BACHELARD 67]), ou didactiques (i. e. dus à l'enseignement), ce sont des déviations du contrat didactique.

Les pédagogies basées sur la reconnaissance et le traitement des erreurs se développent [DUMONT 89b]. Les interventions peuvent s'effectuer :

- en amont (approche appropriée),
- en aval (choix d'activités de remédiation),
- au moment de la production (exploitation en classe).

Il faut noter que la pédagogie proposée par Papert [PAPERT 80] avec LOGO est centrée sur l'erreur (voir II.1.b.2.a, page 33).

V.1.a.3 Objectifs / acteurs du diagnostic

Le diagnostic consiste à essayer de trouver pourquoi une personne s'est trompée dans la réalisation d'une activité, ou même plus généralement, à comprendre comment un comportement a pu être observé : « *le but général est d'obtenir un compte rendu abstrait du comportement d'un dispositif* » [EVERTSZ & ELSOM-COOK 90]. Au niveau cognitif, il s'agit de décrire un processus mental. C'est utile aux :

- Psychologues : rendre opératoires des observations empiriques (méthodes de recherche); en particulier, l'étude fine du diagnostic est nécessaire pour proposer et évaluer des théories de l'apprentissage.
- Enseignants ou didacticiens : évaluer les connaissances, l'absence de connaissances ou les mauvaises connaissances des apprenants.
- Informaticiens : construire des interfaces qui s'adaptent à l'utilisateur, ou des modèles de l'élève dans un tuteur intelligent ou dans un système d'aide.

D'une manière plus précise, dans le cadre de la composante de modélisation de l'apprenant dans un tuteur ou un logiciel d'E. I. A. O., un diagnostic minimal est nécessaire pour (voir [SELF 87], [NICAUD & VIVET 88]) :

- générer ou sélectionner des exercices (problème de l'avancement ou de la gestion du curriculum);
- choisir des modes d'exposition ou de travail;
- faire un suivi pas à pas de l'élève pour localiser l'erreur en situation;
- donner des conseils non sollicités (intervention);
- adapter les explications et les diverses aides disponibles;
- générer des contre-exemples (voir par exemple [EVERTSZ 89] et le système PG génération de contre-exemples sur la soustraction des fractions), ou des exemples discriminants ('critical problems') pour affiner le diagnostic lui-même ([EVERTSZ & ELSOM-COOK 90]).

Pour les enseignants, la collecte de catalogues d'erreurs peut leur fournir un outil d'aide (travaillant soit en ligne, soit en différé), ou d'entraînement (voir BUGGY, V.3.a.2, page 172). Ceci doit aussi conduire à mieux structurer le domaine pour l'apprentissage (élaboration de taxonomies) et à améliorer les types de progression (conditions de félicité [VANLEHN 87]).

On peut noter que tous les acteurs sont intéressés à la fois par les résultats du diagnostic et par sa conception.

V.1.a.4 Problématique générale

Le diagnostic consiste souvent à chercher POURQUOI quelqu'un a fait quelque chose en s'appuyant sur son comportement observable, i. e. COMMENT il l'a fait. Ainsi, si on prend

une analogie médicale, la réponse ou le résultat correspond aux symptômes, le 'comment' à la maladie, et le 'pourquoi' à l'origine de la maladie.

Remarque : cette analogie peut être poussée assez loin :

- des mêmes symptômes peuvent correspondre à plusieurs maladies (le diagnostic nécessite alors des examens complémentaires),
- des maladies différentes peuvent être associées à des causes identiques, tandis que pour des individus différents, les mêmes causes peuvent conduire à des maladies distinctes.
- les thérapeutiques uniquement associées aux maladies sans s'occuper des causes conduisent souvent à des rechutes,
- etc.

Certaines théories s'opposent cependant à cette analogie en considérant que le fait de produire une erreur correspond plutôt au comportement normal, tout au moins dans les phases d'apprentissage. Les deux points de vue sont peut être conciliables : les maladies infantiles bénignes permettent le développement des défenses de l'organisme.

La recherche du 'pourquoi' suppose beaucoup d'hypothèses sur les connaissances réelles d'un élève et leur activation ou non dans un contexte précis. La source principale d'erreur peut d'ailleurs se situer hors du champ didactique : pas de sens attaché à une activité, désintérêt général, etc.

Pour éviter des oppositions inextricables et non productives en considérant le problème du diagnostic dans une trop grande généralité, il est préférable de le décomposer en plusieurs tâches complémentaires :

1. chercher le 'comment',
2. associer le 'comment' au 'pourquoi',
3. traiter la finalité du diagnostic.

La première phase est celle qui est la plus «automatisable» et correspond à trouver le cheminement suivi par un élève pour résoudre un problème. C'est ce type de travail qui nous concerne en priorité ici. Il faut noter qu'en effectuant cette recherche de chemin, on s'éloigne du domaine traité par l'élève, ce qui permet un traitement plutôt syntaxique dont on essaye d'atteindre les limites. En effet, il y a en pratique un côté nécessairement exploratoire et flou de cette démarche dont la finalité est incomplètement définie. Ne sachant pas à l'avance jusqu'où pourra aller le système de diagnostic, on est bien en peine de préciser son futur cadre d'utilisation (constitution d'un catalogue, diagnostic en ligne, entraînement, modélisation dynamique, etc.).

Cette impossibilité à fixer a priori les limites des chemins trouvables par la machine dans la recherche de 'comment' conduit à aller le plus loin possible sans trop structurer le domaine concerné. En particulier, on n'incorpore pas les traces pouvant guider la résolution (au moins dans un premier temps). Aucune modélisation fine de la connaissance n'est ainsi nécessaire dans la phase initiale (elle sera indispensable pour traiter le 'pourquoi').

Remarque : la décomposition précédente n'est légitime que si on accepte la théorie des erreurs systématiques, garantissant l'établissement d'un lien entre les erreurs procédurales faites et les connaissances, bonnes ou mauvaises, d'un sujet. Ceci a pu être vérifié pour de jeunes enfants (voir BUGGY [BURTON & BROWN 78], PIXIE/LMS [SLEEMAN

82]) dans des activités mathématiques; on suppose que cela peut s'étendre à d'autres domaines.

Il faut ensuite pouvoir traiter le point 2, c'est-à-dire réaliser l'articulation avec les connaissances, ce qui introduit de nouveaux problèmes :

- migration des bugs (voir REPAIR THEORY [BROWN & VANLEHN 82]), la même incompréhension peut conduire à des manifestations différentes;
- phénomènes de bruits (erreurs «accidentelles»).
- instabilité des 'bugs' (voir [HENNESSY 90]).

Les 3 phases mentionnées ne sont pas indépendantes. En particulier, des recherches didactiques sur le domaine peuvent fournir un catalogue de comportements erronés, ainsi que des déviations caractéristiques intéressantes à contrôler. Ces résultats peuvent permettre de trouver des heuristiques pour limiter la recherche ou de concevoir des environnements mixtes prenant en charge une partie de la résolution, assurant un meilleur filtrage des traces.

La distinction entre 'comment' et 'pourquoi' n'est pas forcément aussi tranchée. Si on considère qu'un bug est une règle interne qui sous-tend le comportement, l'erreur n'étant qu'une manifestation externe [HENNESSY 90], l'analyse du comportement en termes de bugs est déjà une première incursion dans le domaine de l'explication. L'articulation entre le 'comment' et le 'pourquoi' peut ainsi être vue comme la description de plus en plus fine d'un comportement.

Le type d'utilisation du diagnostic conditionne ce que l'on cherche (erreurs caractéristiques, réalisation d'un catalogue, traces à constituer). De plus, on ne prétend pas pouvoir découvrir complètement ce qui se passe dans la tête d'un sujet, mais avoir une représentation suffisamment fidèle pour qu'elle soit opératoire vis-à-vis du but poursuivi. Ce but donne ainsi un cadre et une sorte de filtre sur le modèle recherché.

Remarque : certains chercheurs pensent que la plupart des erreurs sont dues à un désintérêt des élèves et à une absence totale de sens attaché aux activités menées, que dans ce cadre leurs productions sont quasiment aléatoires et que c'est un jeu un peu vain de vouloir y découvrir une logique, mieux vaut prévenir que guérir [LACOMBE 88a]. Il semble cependant que cette position extrême oublie de faire intervenir le contexte scolaire comme élément d'analyse des comportements, l'enfant cherchant normalement à répondre correctement à ce qui lui est demandé. Les connaissances générales des élèves interviennent de façon déterminante dans ce qu'ils produisent, et ces productions sont des indices importants sur leur état de connaissance. La part d'aléatoire correspond plus à des phénomènes d'instabilité déjà évoqués, dont on connaît encore mal l'ampleur.

" The maxim 'What holds today will hold tomorrow...', is implicit in the process of student modelling. On the face of it, it is valid to assume that the student will always respond in the same way to a given situation. The twist in this assumption is that the student is never confronted with the same situation. " [EVERTSZ & ELSOM-COOK 90]

V.1.b Modélisation d'un utilisateur

V.1.b.1 Historique

Le besoin de disposer d'une représentation des connaissances d'un apprenant s'est vite imposé pour la réalisation de tuteurs capables d'adapter leurs stratégies tutorielles ([HARTLEY & SLEEMAN 73] et [SELF 74]).

Plusieurs types de modèles ont été construits :

- **modèle d'expertise partielle ou de superposition** ('overlay' model) [CARR & GOLDSTEIN 77]. La connaissance de l'élève est considérée comme un sous-ensemble de celle de l'expert.
- **modèles différentiels** [BROWN & BURTON 78] incorporant des connaissances erronées, qui sont des perturbations des connaissances expertes.
- **modèles du fonctionnement cognitif** [PALIES 88] : modèle prévisionnel du comportement d'un élève en situation d'apprentissage. Il contient non seulement l'ensemble des connaissances mises en jeu pour résoudre un problème, mais aussi la représentation mentale que se fait l'apprenant du problème. Cette dernière approche considère les connaissances de l'élève comme des conceptualisations du domaine fondamentalement différentes de celles de l'expert. Elle permet de réellement prendre en compte les erreurs profondes ('misconceptions') d'un apprenant.

Ce dernier type de modèle répond à une objection de Self et O'Shea [O'SHEA & SELF 83] sur le fait que les modélisations de l'élève tendent à opérer à un mauvais niveau : ils fournissent des informations sur les tentatives d'un élève pour résoudre un problème spécifique, mais pas directement sur sa compréhension des aptitudes générales mises en jeu.

V.1.b.2 La classification de Kurt VanLehn

Kurt VanLen [VANLEHN 88] explique comment classifier les différents modèles élèves existant dans un espace à trois dimensions. Il tient compte de trois types de paramètres :

1. Les 'entrées', i. e. les renseignements utilisables par le module diagnostic :
   - l'ensemble des états mentaux ('mental states'), i. e., toute l'activité physique et mentale est utile,
   - des états intermédiaires, i. e., toute l'activité physique observable est utile,
   - l'état final uniquement, i. e., la réponse.
2. Les connaissances utilisées par le module expert :
   - procédurale plate, pas de sous-buts,
   - procédurale hiérarchique, i. e., avec des sous-buts,
   - déclarative.
3. Les différences élève-expert :
   - expertise partielle ou superposition ('overlay'),
   - modèle différentiel avec une librairie prédéfinie de 'bugs',

- modèle différentiel avec une librairie prédéfinie de 'bugs' partiels qui sont assemblés dynamiquement pour rendre compte du comportement de l'élève.

Connaissance 'Entrées'	Procédurale plate	Procédurale hiérarchique	Déclarative
Etats mentaux		** Kimball CALCULUS ** LISP TUTOR ** GEOMETRY TUTOR	GUIDON
Etats intermédiaires	WEST WUSOR	** MACSYMA Advisor ** SPADE ** IMAGE	* SCHOLAR * WHY * GUIDON
Etat final	** LMS ** PIXIE ** ACM	* BUGGY * DEBUGGY * IDEBUGGY	* MENO * PROUST

** Librairie de bugs partiels
* Librairie de bugs
Overlay

Cette classification n'utilise pas les connaissances de type qualitatif, et ne rend pas compte des modèles conceptuels qui ne sont pas encore intégrés dans des tuteurs.

V.1.b.3 Les perspectives actuelles

On cherche maintenant à intégrer les connaissances profondes de l'élève dans le modèle, dans un système global de croyances. Croyances du sujet dans le domaine considéré, mais aussi croyances du sujet sur sa propre connaissance.

D'après du Boulay et Sloman [du BOULAY & SLOMAN 88], les élèves devraient être modélisés comme des êtres affectifs, avec des motivations à la fois intellectuelles et non-intellectuelles et des capacités à jouer un rôle dans l'apprentissage. Les tuteurs intelligents devraient prendre en compte les croyances des élèves sur leur propre savoir (croyances qui peuvent être fausses) et pas seulement la connaissance du domaine lui-même : une croyance erronée d'un élève sur ce qu'il comprend constitue un obstacle très important à l'apprentissage.

On passe ainsi d'un modèle élève, plutôt de type statique, à un modèle apprenant de type dynamique, qui doit représenter l'évolution dans la phase d'apprentissage. Gilmore et Self [GILMORE & SELF 88] proposent ainsi un nouveau rôle pour un modèle d'élève : une description psychologiquement crédible d'un partenaire collaborant. Ceci conduit à l'utilisation de techniques avancées de l'IA comme l'ASA (apprentissage symbolique automatique [KODRATOFF 86]) et les logiques floues ou les logiques de la croyance (voir par exemple TAPS [DERRY & Al. 89a]).

Parallèlement et paradoxalement, les difficultés de mise à jour et d'utilisation d'un tel type de modèle invitent les chercheurs à le simplifier. La notion de collaboration entre un système éducatif informatique et un utilisateur (voir collaboration) devrait faciliter le développement d'un tel système en réduisant la nécessité d'une complète précision de la représentation du domaine et du modèle élève [CUMMINGS & SELF 89].

Les liens entre le modèle élève et les autres modules d'un système d'EIAO induisent le type que l'on choisit (voir [OHLSSON 86]), de même que la perspective d'apprentissage retenu,

construire des systèmes qui autorisent un transfert contrôlé des connaissances d'un expert artificiel à un apprenant humain [COSTA & Al. 87].

Une autre position extrême est celle d'Anderson qui n'est pas partisan des modélisations de l'élève [ANDERSON 89], prônant la faible importance d'exprimer un diagnostic cognitif sur les erreurs, dans de nombreux cas la rétroaction du tuteur devant être minimale, voire inexistante.

On peut remarquer que dans certaines produits expliqués dans cette thèse (LYRE III.2.b.3, page 95, ARRIA IV, ...), aucune modélisation de l'élève n'est incluse et ceci non pour des raisons techniques mais pour des raisons didactiques : on a encore très peu d'idées sur le mode de raisonnement des élèves dans des domaines un peu complexes, et on sait encore moins adapter les interventions à des erreurs dont on ignore complètement la genèse. En plus, dans le cas de LYRE, on ne dispose pas de résolveur, et on est donc dans l'impossibilité d'établir un diagnostic, à moins d'introduire un QCM (questionnaire à choix multiples) plus ou moins camouflé!

V.1.c Techniques de diagnostic

V.1.c.1 La classification de Kurt VanLehn :

La classification précédente introduite par Kurt VanLehn sur les modèles élève (V.1.b.2, page 152) est associée directement à des techniques de diagnostic [VANLEHN 88].

Connaissance 'Entrées'	Procédurale plate	Procédurale hiérarchique	Déclarative
Etats mentaux		'Model Tracing'	
Etats intermédiaires	'Issue Tracing'	Reconnaissance de plans	Système Expert
Etat final	Rech. chemin Condition Induction	Arbre de décision Génération/test Interactif	Génération/ Test

Le 'modèle de trace' (model tracing) suppose que deux états consécutifs dans la résolution du problème par l'élève peuvent être reliés par une règle (complétude du modèle).

Ce tableau montre les diverses techniques de diagnostic utilisées suivant les modèles de connaissances formalisées et les types de traces gardées. Dans notre travail, on se contente dans un premier temps de la réponse finale, c'est-à-dire qu'on ne dispose ni de résultats intermédiaires, ni de possibilité de conduire un dialogue explicatif. On implante ainsi un processus de génération/test qui doit être facilement évolutif.

Si on ne connaît que l'état final, on est face à un problème de recherche de chemin. Avec la connaissance d'états intermédiaires, on est souvent plus proche de la reconnaissance de plans.

Nous allons passer en revue quelques programmes de diagnostic.

V.1.c.2 La reconnaissance de plans :

La reconnaissance de plan correspond au processus qui consiste à inférer un arbre plan quand on ne connaît que les feuilles. C'est similaire, au niveau du calcul, d'analyser une chaîne dans une grammaire hors contexte ('context free') un arbre d'analyse est construit, arbre dont les feuilles sont les éléments de la chaîne (*" the student's actions are the words and his planning methods are the rules of grammar "*.)

La tâche prototypique consiste à essayer de reconnaître la structure d'une séquence observée pour discerner le but poursuivi. Il peut d'ailleurs y avoir plusieurs buts, voire même aucun.

On distingue deux techniques principales [ROSS & LEWIS 87] :

- générer des plans et vérifier s'ils correspondent,
- analyser la séquence suivant une grammaire.

On peut citer quelques papiers généraux [JONES & Al. 88] [WOODROFFE 88], et des systèmes précurseurs : NOAH [SACERDOTTI 77], BELIEVER [SCHMIT & Al. 78], MACSYMA ADVISOR [GENESERETH 82], POISE [CARVER 84].

L'approche de T. Walsh dans PLATO [WALSH 88] est intéressante, car elle représente explicitement des heuristiques dans l'expertise du domaine, heuristiques qui guident la reconnaissance des plans de l'élève. Il faut remarquer que l'extension d'une interaction de type 'model tracing' implique l'utilisation de techniques de reconnaissance de plans. (voir le système ALADDIN [SKWARECKI 88] pour une nouvelle version de LISP-TUTOR et Hyper-Arria, IV.3.a.2).

V.1.c.3 ACM /DPF

L'approche diagnostique proposée par Langley et Ohlsson élimine le besoin d'un catalogue de bugs, en utilisant des procédures classiques de recherche d'un chemin dans un espace de problème associées à des techniques d'apprentissage automatique. Elle est utilisée dans ACM (Automatic Cognitive Modeling) [LANGLEY & OHLSSON 84] et DPF (Diagnostic Path Finder) [OHLSSON & LANGLEY 88].

V.1.c.3.a Méthodologie générale

Ils partent d'une analyse proche de celle de Newell et Simon [NEWELL & SIMON 72] On dispose d'un espace de problème, un état initial, un état final (un critère de terminaison) et un ensemble d'opérateurs : le diagnostic (le comment) revient à trouver un chemin dans cet espace :

- état initial : concepts (objets, prédicats, etc.),
- état final : réponse (critère de transformation),
- opérateurs : actions (changement d'état).

On peut remarquer que les problèmes d'interprétation et de diagnostic sont deux problèmes de résolution duaux :

- l'interprétation part d'une base de connaissances et d'un problème et doit fournir une solution,
- le diagnostic part d'un problème et d'une solution et doit induire la base de connaissances.

La méthodologie générale de recherche de diagnostic comporte trois étapes :

1. Trouver l'espace de problème (ce qui est souvent loin d'être trivial).
2. Chercher un cheminement aboutissant à une solution : cela consiste à trouver comment la solution a été engendrée.
3. Induire des règles générales de résolution, i. e. passer du comment au pourquoi.

Cette troisième phase utilise les techniques de l'apprentissage automatique : généralisation, induction et "chunking" (groupement de pas élémentaires en pas plus importants). Elle suppose d'avoir un certain nombre d'exemples à disposition.

V.1.c.3.b Problèmes liés à la recherche

La recherche effectuée dans l'espace de problème peut soit être complète, c'est-à-dire exhaustive ou partielle, i. e. sélective. La première paraît plus séduisante, elle est malheureusement impraticable à cause des risques d'explosion combinatoire. Ceci oblige donc à se contenter d'une recherche limitée et à définir des critères ou une fonction d'évaluation. Ils sont basés sur une idée générale de plausibilité psychologique. On distingue les critères absolus qui permettent de rejeter définitivement certaines branches et les critères relatifs influant sur les choix des branches à explorer en priorité. On reprend ici la taxonomie introduite par Ohlsson (DPF) :

1. Les critères absolus :
   - fermeture causale : pas d'intervention divine, c'est-à-dire, élimination de ce qui ne conserve pas de lien causal avec le problème, rejet d'hypothèses trop farfelues.
   - utilité pour le but poursuivi : rejet des pas superflus.
   - pas de duplication : on ne cherche pas à nouveau un résultat déjà obtenu.
2. Les critères relatifs :
   - chargement de la mémoire : à minimiser;
   - satisfaire le plus de sous-buts;
   - critère de productivité : rapport entre le nombre de sous-buts et le nombre d'opérateurs utilisés dans le chemin;
   - minimum d'erreurs;
   - longueur minimale : critère d'économie, on privilégie les chemins les plus courts.

V.1.c.3.c Analyse de cette approche

DPF a été testé sur la soustraction [OHLSSON & LANGLEY 88] et donne de bons résultats. La généralisation à des domaines plus complexes ne semble cependant pas immédiate. En particulier, la détermination de l'ensemble des opérateurs peut être une forme déguisée d'un embryon de catalogue d'erreurs, et la description des états conditionne ce que l'on

peut découvrir. Fournir au programme un ensemble initial d'opérateurs suppose une théorie implicite des types de 'bugs' qui peuvent se manifester. Sans une telle théorie informelle, l'explosion combinatoire de l'espace de recherche ne peut être évitée, laissant ACM face à une tâche sans espoir [EVERTSZ & ELSOM-COOK 90].

Le travail sur BADAUD reprend en partie l'analyse précédente en se limitant aux deux premières étapes, la dernière étape n'étant pas automatisée.

D'autres systèmes utilisent des techniques d'apprentissage automatique pour la création des modèles de l'élève. On peut citer par exemple ALCHEMIST ([COSTA & Al. 87],[BENTO & COSTA 88]). Les erreurs de l'apprenant sont considérées comme l'utilisation de conditions erronées conduisant à sélectionner un opérateur inadéquat. Les circonstances de choix forment les exemples positifs et négatifs à partir desquels se construit l'apprentissage. De la même manière, Frasson et de la Passardière veulent construire des modèles hypothétiques intégrant le contexte dans lequel les connaissances ont été acquises [FRASSON & de la PASSARDIERE 89].

V.1.c.4 Le système de Y. M. Visetti

Le travail de Y. M. Visetti [VISETTI 86] concerne les calculs sur les fractions. Il s'attache à reconstituer un plan mis en oeuvre par un élève dans la conduite d'un calcul pour mettre à jour un modèle de cet élève. Le système fait appel à la notion de plan en considérant un calcul comme une suite de tâches. Il distingue les tâches générales, i. e. celles qui permettent d'organiser un calcul, de sélectionner les sous-expressions à traiter, et les primitives dont l'exécution est assurée par une seule procédure (Evaluer, simplifier sont des tâches primitives, ce qui se justifie sur le domaine des fractions avec des petits nombres, mais peut poser des problèmes dans des cadres plus complexes, voir VI.4.a, page 199).

« Le modèle de l'étudiant déclare pour chaque tâche primitive un ensemble de noms de procédure susceptibles a priori d'intervenir dans sa réalisation. Le diagnostic consiste en particulier à classer ces procédures 'candidates' suivant leur fréquence d'apparition; cette classification fonde la description des compétences relativement à chacune des tâches élémentaires. »

Il s'appuie sur la notion de contexte, vu comme un couple <tâche,situation>, et des relations d'ordre (généralisation, particularisation) entre ces contextes. Les erreurs sont ainsi des connaissances déviées intervenant dans des contextes formels. Pour éviter une trop grande multiplicité de déclarations de procédures erronées, il est amené à représenter des corrélations liant les comportements observés dans différents contextes.

Le but est de faire un 'modéliseur' de l'élève qui peut être intégré dans un système d'EIAO à l'architecture classique. Le cas des ambiguïtés (plusieurs possibilités pour effectuer un pas de calcul) est levé par des critères de plausibilité et un dialogue avec l'élève.

V.1.c.5 TAPS

Dans le cadre du développement de TAPS (tutoriel sur les 'word problems', voir II.2.b.3), S. Derry [DERRY & Al. 89a] distinguent deux types de diagnostic :

- le diagnostic local basé sur la performance courante de l'élève, surtout lié à l'utilisation des schémas, (Vergnaud, Greeno,...),

- le diagnostic global qui doit faciliter les décisions globales du tuteur.

Ils énumèrent trois types de difficultés principales :

1. Comment rendre des jugements valides en dépit des 'bruits' obtenus?
2. Comment maintenir et mettre à jour des modèles individuels qui puissent être détaillés et accessibles pour le tuteur?
3. Comment utiliser ces modèles pour guider les interventions aussi bien au niveau global (curriculum) qu'au niveau local (niveau opportuniste)?

Leur réponse consiste à implanter des modèles flous : "*fuzzy pattern matching, fuzzy temporal relational database* ", i. e. penser en des termes linguistiques imprécis de type humain.

Au niveau purement diagnostic [DERRY & Al. 89b], ils s'appuient sur un catalogue représentant 98% des erreurs faites, et une méthode de déviation utilisant une représentation intermédiaire ("graphic reification").

D'après les auteurs, cette approche est meilleure que celle de BUGGY :

- détection en ligne,
- nombre d'erreurs pas trop important (34), mais une infinité de procédures buggées pour représenter ces erreurs,
- liaison facile avec les interventions tutorielles,
- facilitation de la modélisation de l'élève.

V.1.c.6 Simplifier la recherche du diagnostic :

Le type de méthodes précédentes (voir TAPS), cherchant à simplifier le diagnostic en l'associant plus étroitement aux interventions tutorielles et utilisant des environnements adaptés permettant un premier filtrage, devient tout à fait classique.

D'autres systèmes adoptent cette approche. Ainsi, l'étude sur la soustraction de Young et O'Shea [YOUNG et O'SHEA 81] souligne l'existence de 15 erreurs fondamentales et le fait que les multiples bugs de BUGGY ne sont pas assez profonds, et souvent trop similaires, ce qui les rend inutilisables pour la remédiation.

Les mêmes réserves sont faites par R. Nicolson, qui construit le système SUMMIT [NICOLSON 88] en ne tenant compte que des erreurs les plus fréquentes (8 erreurs d'addition couvrent 92,3% des cas, 8 erreurs en multiplication couvrent 91,4% des cas, voir II.2.b.1).

Le debugger automatique CHIRON [SACK 88] est une réétude de PROUST [JOHNSON 85] qui s'appuie sur des considérations similaires :

- 70% des erreurs sont dues à :
  - des détails (syntaxe, substitution d'opérateurs, etc.),
  - des mauvaises coordinations (accoler les bouts ensembles, voir IV.3, page 135) (intra- i. e. dans un seul plan, inter- dans plusieurs plans);
- les faiblesses des debuggers utilisant des 'mal-rules' :

- tous les bugs sont égaux : tous représentés comme fausses règles, aussi bien ceux qui sont cognitivement plausibles que les autres. L'introduction d'un nouveau bug implique la création d'une nouvelle règle. Il faudrait pouvoir représenter différemment les bugs fréquents et les bugs rares.
- les 'Mal-rules' sont trop spécifiques : pas de super bug-rules,
- les bugs dérivés ne sont pas représentés,
- les parties de programme sont supposées indépendantes. (remarques à relier à l'analyse de Hennessy sur les insuffisances de l'approche BUGGY [HENNESSY 90]).

CHIRON travaille avec des plans définis dans un langage de frames (PANGLE), et des généralisations de 'fragments' avec des hiérarchies prédéfinies (instance / classe, partie / tout). Il peut disposer aussi de plusieurs représentations (tranches) du même programme (voir représentations).

J. Self, fervent partisan de la mise en place de modèles élèves performants dans les logiciels d'EIAO, donne quelques indications et réflexions pour les construire d'une manière effective [SELF 88] :

- Eviter de deviner, demander à l'élève ce que vous voulez savoir,
- Ne pas diagnostiquer ce qu'on ne peut traiter,
- Tenir compte des croyances de l'élève, ne pas les cataloguer comme bugs : « *les modèles élèves sont destinés à décrire non ce que les élèves savent, mais ce qu'ils croient* »,
- Ne pas feindre l'omniscience, adopter un rôle de collaborateur «faillible».

Self reconnaît aussi l'utilité du module environnement qui permet d'éviter des recherches trop complexes. Ainsi, la déclaration des buts facilite cette recherche (EPIC [TWIDALE 89]) et n'est pas dénuée d'intérêt sur le plan didactique. Il doit ainsi être possible, pour un tuteur intelligent, de donner des conseils stratégiques à partir de la mise en évidence des propriétés structurelles de l'espace de recherche (réflexion sur ALGEBRALAND de Foss).

Dans le cadre des systèmes d'EIAO, deux techniques principales permettent de simplifier considérablement la recherche du diagnostic :

- faire déclarer le but ou les intentions,
- utiliser des représentations intermédiaires.

BRIDGE [BONAR & CUNNINGHAM 88b] cumule ces deux approches. C'est en quelque sorte une méthode duale de celle de PROUST qui utilise une base d'opérateurs corrects et buggés et qui est très coûteuse (temps de recherche, connaissances ingénieur). BRIDGE est plus un environnement didactique qu'un debugger.

V.1.d Problèmes généraux du diagnostic

La conception d'un système de diagnostic repose sur la détermination des bugs élémentaires et l'identification des causes profondes des erreurs constatées.

V.1.d.1 Comment trouver les 'bugs'?

Les bugs peuvent être trouvés :

- à partir d'une étude disponible dans le domaine (littérature),
- par l'analyse du comportement des élèves (avec divers protocoles),
- grâce à un modèle prédictif s'appuyant sur une théorie de l'apprentissage dans le domaine concerné.

L'analyse préalable des erreurs est un processus coûteux [HENNESSY & Al. 89] qui n'est jamais totalement fini (une nouvelle erreur peut toujours intervenir). Les méthodes de perturbation (comme BADAUD, ou BOOK [TAKEUCHI & OTSUKI 87]) génèrant tous les chemins possibles conduisant à la réponse de l'élève n'échappent pas aux problèmes de complétude (voir complétude). Leur rôle s'apparente plus à celui de systèmes d'aide à la décision (idée de 'diagnostic consultant').

Les modèles de comportement plus profonds facilitent les explications et les suggestions de remédiation et permettent une génération des bugs. Yazdani [YAZDANI 89] donne l'exemple de son travail sur l'enseignement de l'anglais comme langue étrangère. Des correspondances entre la grammaire anglaise et celle de la langue maternelle des apprenants permet une meilleure prise en compte des erreurs grammaticales que les larges taxonomies des erreurs courantes utilisées dans les systèmes précédents.

La spécificité de cet exemple montre bien que, si cette méthode est sans conteste plus satisfaisante, son applicabilité à d'autres domaines est loin d'être immédiate.

V.1.d.2 Comment trouver les causes profondes?

Les erreurs proviennent essentiellement de déviations sur des propriétés de surface (e. g. syntaxiques en mathématiques) et de préconceptions erronées. La recherche des sources des erreurs et de leur mode de génération en liaison avec les connaissances des sujets (e. g. REPAIR), est encore très embryonnaire [HENNESSY 90]. L'analyse des préconceptions rejoint la recherche des bugs.

Les réflexions sur les causes des erreurs rejoignent d'autres préoccupations de cette thèse :

- les 'bugs' interviennent comme le résultat de l'application entre les principes ('conceptual') et la connaissance procédurale. Il faudrait aider l'élève à établir des liens entre les procédures et les concepts : hypertexte réponse aux problèmes de sens).
- Collins and Goldin (1979) (cité dans [HENNESSY 90]) : " *The availability of multiple viewpoints raises the possibility that some errors may turn out to reflect the fact that some perspective is missing in the student's knowledge. Alternatively, some bugs might be described as a failure to integrate several different viewpoints, a failure to recognise contradictions between two different perspectives or to update one viewpoint when relevant changes are made in another.* " (voir la notion de point de vue, III.2.b.5, page 98). Ceci implique que la représentation de la connaissance doit être multi-dimensionnelle.

Chapitre V.2

Le programme BADAUD

V.2.a Fonctionnement général

BADAUD est un outil de diagnostic comportemental pour des tâches procédurales qui s'attache à fournir, connaissant le résultat final et éventuellement certains calculs intermédiaires, l'ensemble des cheminements plausibles conduisant à ce résultat. Il ne s'appuie sur aucun modèle d'élève et ne tient pas compte du contexte pour privilégier un chemin parmi d'autres. Il ne cherche pas à être un outil prédictif du comportement d'un élève donné contrairement au système de Visetti (V.1.c.4, page 157) ou à DPF (Diagnostic Path Finder, V.1.c.3, page 155). On reprend néanmoins une partie de l'analyse de DPF : espace de problème, état initial, état final (critère de terminaison) et un ensemble d'opérateurs; le diagnostic revient à trouver un chemin dans cet espace.

Pour effectuer une telle recherche, la tâche étudiée doit satisfaire à plusieurs conditions concernant la réponse :

- elle ne doit pas être donnée avec l'énoncé du problème;
- elle doit être choisie parmi un large éventail possible;
- elle doit posséder une forme de structure interne.

C'est le cas dans les exemples traités (calcul sur les fractions, résolution de problèmes de chimie).

La première étape consiste à créer un résolveur, puisque la machine doit être capable de résoudre le problème en utilisant des connaissances correctes ou erronées. Ceci doit respecter différentes contraintes :

- un résolveur le moins «câblé» possible,
- catalogue de comportements erronés afin de pouvoir :
  - ajouter de fausses connaissances,
  - perturber la stratégie de résolution.

Cette étape peut se traduire par une forme de bricolage car elle associe des éléments disparates. L'écriture d'un résolveur aisément perturbable est difficile, puisqu'on ne sait pas à

l'avance ce qui est susceptible d'être modifié. La seule possibilité consiste à essayer d'implanter les choses de la manière la plus déclarative possible.

Remarque : on retrouve le conflit entre le comment et le pourquoi. En effet, les études didactiques cherchent un lien entre un contexte particulier et un type d'erreur. Ainsi, en ce qui concerne l'addition des fractions : 2/3 + 3/5 → 2/5 Les deux 3 rapprochés induisent une simplification abusive. Ceci ne sert pas directement un système de diagnostic devant trouver TOUTES les erreurs possibles, non directement les expliquer (a contrario de DPF ou du système de Visetti). Le passage du local au global engendre aussi des explosions amenant aussi des comportements très improbables. Nous verrons plus loin comment éliminer une partie de ces cheminements superflus.

On a en fait 2 possibilités :

- suivre pas à pas la ou les résolutions habituelles : il faut pouvoir prévoir toutes les stratégies que peuvent adopter les élèves. Les erreurs sont ainsi bien liées à une étape de résolution.
- résoudre d'une manière non classique, en essayant d'intégrer les erreurs a priori possibles. Il faut ensuite reconstruire un cheminement plausible. En effet, la combinaison d'erreurs observées ne donne pas forcément des résultats cohérents.

C'est cette deuxième solution que l'on retient, la première ayant le désavantage de nécessiter la connaissance des méthodes utilisées par les élèves. Les plans effectivement utilisés par ceux-ci pourront être réintroduits dans une phase ultérieure pour affiner la recherche du diagnostic.

La section suivante décrit un premier essai d'implantation de BADAUD dans le domaine du calcul sur les fractions. Le travail effectué sur la résolution des problèmes de chimie n'est pas relaté dans le présent document. La dernière section explique le fonctionnement du programme générique bâti à partir de ses deux implantations, ainsi que son insertion dans un travail complet de recherche de diagnostic (BADAUD n'est qu'un élément, un outil qui s'insère dans une démarche globale).

V.2.b Etude des fractions

Cette recherche a été menée à l'INRP (Institut National de Recherche Pédagogique), par une équipe dont la responsabilité était confiée à Bernard Dumont et comprenant J. F. Boudinot, E. Bruillard, J. P. Drouhard, B. Grugeon, Y. Paquelier et C. Terlon. Elle s'appuie sur une première étude sur les erreurs dans les opérations sur les fractions menée à l'Université Paris VII [DUMONT 84] et une large enquête menée durant l'année scolaire 1987-1988. L'objectif de cette deuxième enquête était de rechercher systématiquement toutes les erreurs produites, à la fois dans les opérations sur les fractions et sur les simplifications et de vérifier l'existence des corrélations entre les erreurs elles-même et la présentation faite par l'enseignant de mathématiques (un compte rendu est en cours de parution, voir aussi [DUMONT 90]).

Ce qui est relaté dans la suite ne concerne que mon travail sur le système de diagnostic bâti à partir des erreurs classées par les didacticiens.

Rappelons qu'un travail sur le diagnostic d'erreurs dans le calcul des fractions a été effectué par Y. M. Visetti [VISETTI 86] (V.1.c.4, page 157), et que d'autres équipes travaillent à la

réalisation de tuteurs intelligents dans ce même domaine ([NWANA & COXHEAD 88] et [KONDO & Al. 90], II.2.b.4).

V.2.b.1 Sommes et différences de fractions

A partir des résultats de l'enquête sur les sommes et différences de fractions, divers systèmes ont été mis au point. Je ne parlerai ici que du système élaboré en Prolog. Ce dernier a d'ailleurs montré une meilleure adéquation de Prolog, pour une telle réalisation, par rapport à un générateur de systèmes experts (en effet, Prolog assure une plus grande flexibilité, permet directement l'utilisation conjointe de calculs et de raisonnements et est particulièrement bien adapté pour la recherche dans des arbres!).

Les objets à manipuler sont des objets mathématiques courants : des nombres, des fractions et des opérations sur ces nombres et ces fractions. L'énoncé du problème se compose de 2 fractions et un opérateur (+, - ou *). Le résultat final est une fraction. Les opérateurs de transformation utilisés, outre les opérations sur les entiers, sont la réduction au même dénominateur de 2 fractions et la simplification d'une fraction.

Un recensement complet des erreurs sur les sommes et différences de fractions trouvées à partir du dépouillement des enquêtes a été effectué.

On peut les classer en gros en quatre catégories :

- mauvais algorithmes de réduction au même dénominateur,
- fausses simplifications,
- changements d'opérateur,
- erreurs de calcul sur les nombres (ou erreur de recopiage de l'énoncé).

Les dernières erreurs sont évidemment très difficiles à prendre en compte. Dans un premier temps, on ne les a pas intégrées, considérant qu'elles n'étaient pas forcément pertinentes pour le problème traité et que, dans un environnement de travail intégrant une calculette, elles devraient s'estomper (sans pour autant disparaître complètement). Elles présentent néanmoins certaines régularités (erreur d'une dizaine ou d'une unité), mais amènent à ce stade une explosion combinatoire certaine pour un intérêt très minime. Elles ont été reprises en compte dans l'étude spécifique sur la simplification. Une première approche amène à les considérer comme des erreurs non systématiques qui renseignent peu sur la maîtrise d'un élève vis-à-vis des calculs sur les fractions. Cette affirmation sera nuancée plus loin (V.2.b.2.a, page 164). En tout état de cause, dans le cadre d'un système diagnostic intégré à un environnement tutoriel, il est toujours possible de demander à l'élève de vérifier ses calculs : ceci devrait suffire dans la majorité des cas à se débarrasser du problème. Les autres erreurs peuvent être prises en compte d'une manière systématique.

Le programme fonctionne aussi bien en test (résultat donné, chercher un chemin) qu'en génération (donner tous les résultats possibles et la façon de les obtenir). Il comprend deux ensembles de clauses :

- le premier concerne les diverses réductions au même dénominateur,
- le second concerne les simplifications (à partir d'une factorisation en produit de nombres premiers du numérateur et du dénominateur).

S'ajoutent deux types de perturbation :

- le passage d'un entier à la fraction correspondante :
  N donne N/1 ou quelquefois N/N

- les changements d'opérateur :
  $F1 + F2 \rightarrow F1 * F2$
  $F1 - F2 \rightarrow F1 + F2$
  $F1 - F2 \rightarrow F1 * (-F2)$
  $F1 * F2 \rightarrow F1 + F2$

La première mouture du système correspondant aux caractéristiques décrites ci-dessus a pu être testée sur un grand nombre de cas répertoriés dans les enquêtes et a donné les résultats escomptés. N'ayant pas à faire face à de gros problèmes d'explosion, le système étant suffisamment rapide, une deuxième version, incluant de nombreuses déviations dans la simplification, a été implantée.

V.2.b.2 Etude de la simplification

V.2.b.2.a Fonctionnement général

Une étude spécifique a été menée sur les simplifications de fractions. Les tableaux ci-joints résument différents résultats :

- tableau A : liste des élèves,

- tableau B : liste et résultats des différents exercices proposés,

- tableau C : liste des actions erronées.

Tableau A
Effectifs pour la simplification

niveau	nb élèves	nb classes
6ème	265	11
5ème	403	16
4ème	514	20
3ème	318	13
total	1500	60

Tableau B
Liste des exercices de simplification

	Énoncé	Après calcul	Réponse
a1	42/72		7/12
b1	66/44		3/2
c1	72/42		12/7
d1	77/44		7/4
e1	24/27		8/9
f1	70/105		2/3
g1	27/24		9/8
h1	42/63		2/3
i1	75/45		5/3
j1	14/84		1/6
a2	18/81		2/9
b2	81/18		9/2
c2	30/40		3/4
d2	72/75		24/25
e2	10/30		1/3
f2	75/72		25/24
g2	42/48		7/8
h2	45/75		3/5
i2	63/42		3/2
j2	105/70		3/2
a3	63/42		3/2
b3	42/63		2/3
c3	45/75		3/5
d3	75/45		5/3
e3	75/72		25/24
f3	81/18		9/2
g3	66/44		3/2
h3	77/44		7/4
i3	72/75		24/25
j3	18/81		2/9
a4	$8*6+22/105$	70/105	2/3
b4	$9*8+33/70$	105/70	3/2
c4	$4*5+4/27$	24/27	8/9
d4	$3*7+9/40$	30/40	3/4
e4	$9*5-31/84$	14/84	1/6
f4	$4*6+18/72$	42/72	7/12
g4	$5*9+27/42$	72/42	12/7
h4	$5*5+17/48$	42/48	7/8
i4	$2*7+13/24$	27/24	9/8
j4	$7*7-39/30$	10/30	1/3

Tableau C
Etude réalisée par J.P. Drouhard et Y. Paquelier

Liste des actions erronées	Exemples	Codes		
simplif. correcte (incomplète)	$75/45 = 25/15$	sc	364	24,90%
erreur de calcul	$75 = 14 * 5$	ec	359	24,56%
résultat non simplifié	$(23+19)/63 = 42/63$	ns	339	23,19%
simplification euclidienne	$75/45 = 37/22$	se	219	14,98%
simplification hétérogène	$75/45 = 15/5$	sh	164	11,22%
valeur décimale de la fraction	$75/45 = 1,66$	vd	132	9,03%
mystérieux		quid	120	8,21%
simplif. donnant résultat décimal	$75/45 = 8,33/5$	sd	120	8,21%
simplification hétérogène isolée	$81/18 = 9/3$	sh(?,?)	78	5,34%
multiplication (num. et/ou dén.)	$75/45 = 150/90$	m	68	4,65%
simplification décimale par 2	$105/70 = 52,5/35$	sd2	67	4,58%
erreur de calcul dans l'addition	$(23+19)/63 = 52/63$	eca	57	3,90%
simplif. avec inversion num / dén.	$75/45 = 9/15$	inv	48	3,28%
simplif hétérogène (1er facteur : 2)	$18/81 = 2/9$	sh(2,?)	45	3,08%
simplif euclidienne (num.)	$75/45 = 8/5$	sen	44	3,01%
monstre (curiosité non codée)		tera	42	2,87%
erreur dans la décompos en facteurs	$75 = 3 * 3 * 5$	edfp	41	2,80%
erreur de calcul partiel	$(3 * 7 + 9)/40 = 29/40$	ecp	34	2,33%
simplification décimale par 10	$75/45 = 7,5/4,5$	sd10	29	1,98%
erreur de calcul dans *	$(5 * 9 + 27)/42 = (40 + 27)/42$	ecm	28	1,92%
remp. quotient par diviseur	$75/45 = 5/9$	rqdn	28	1,92%
simplif euclidienne (dén.)	$75/45 = 3/2$	sed	28	1,92%
erreur de transcription (recopiage)		et	23	1,57%
simplif hétérogène (2ème facteur :2)	$75/72 = 25/36$	sh(?,2)	22	1,50%
remplac.t du quotient par le divi	$75/45 = 25/3$	rqdd	21	1,44%
Trois simplifications décimales		sd*sd*sd	20	1,37%
simplification par soustraction	$75/45 = 72/42$	ss	20	1,37%
simplification avant effectuation	$(51+26)/44 = 64/22$	sae ou sp	16	1,09%
enchaînement de 3 simplif hétérog	$21/27 = 12/27 = 6/27 = 3/27$	sh*sh*sh	11	0,75%
simplification typographique	$75/45 = 7/4$	st	10	0,68%
barrer un nombre donne "rien"	$9/18 = 0/2$	br	9	0,62%
simplification incomplète en cour	$75/45 = 25/15$	sc?-	7	0,48%
remplacement de la multiplication	$(5 * 9 + 27)/42 = 41/42$	rma	6	0,41%
réécriture de la fraction donnée	$75/45 = 75/45$	rc	5	0,34%
remplac.t de l'addition par la mu	$(23+19)/63 = 437/63$	ram	1	0,07%
simplif incomplète par 11	$44/66 = 4/6$	sc??-	0	0,00%
			1462	100,00%

Ce deuxième tableau amène quelques commentaires. Il a été constitué au fur et à mesure de l'analyse des copies des élèves et les actions sélectionnées répondent à un besoin premier de codage de l'ensemble des erreurs rencontrées pour une évaluation statistique. L'ampleur de la tâche, et le fait que les erreurs ne sont pas connues à l'avance introduisent quelques difficultés dans ce codage.

Tout d'abord, l'aspect incrémental de celui-ci empêche d'assurer une cohérence complète, il faut admettre ainsi un certain pourcentage de flou voire d'erreurs dans le récapitulatif des

résultats (c'est cependant assez minime).

Ensuite, il s'écarte de l'implantation informatique pour différentes raisons :

- codes inutiles (simplification incomplète par 11, simplification incomplète en cours de calcul ou enchaînement d'erreurs élémentaires),
- codes redondants ou des particularisations de codes plus généraux (simplification typographique et simplification euclidienne par 10),
- codes non différentiables dans certains cas sans recours à la copie de l'élève (erreur de calcul, erreur de calcul dans la décomposition en facteurs premiers, simplification hétérogène, ou simplification hétérogène et remplacement du quotient par le diviseur),
- codes inutilisables : soit ininterprétables (quid ou tera), soit impossibles à prendre en compte (erreur de transcription).

Ces écarts sont dus à la fois au mode de production des codes des erreurs, mais aussi aux exploitations différentes qui en sont faites. En particulier, les codes non différentiables sans recours à la copie sont plus proches d'une interprétation de l'erreur que d'une déviation élémentaire dans un calcul. Ces codages mélangent des niveaux d'interprétation différents. Ces quelques remarques ne sont pas là pour déprécier l'énorme travail de dépouillement, mais pour montrer que le processus de création d'un système de diagnostic est essentiellement dialectique et que les listes d'erreurs élémentaires doivent obligatoirement être revues au fur et à mesure de l'implantation d'un système automatique.

Dans ce travail sur la simplification, le traitement des erreurs de calcul a été intégré. Certaines régularités constatées au niveau de ces erreurs permettent d'en prendre en compte une bonne partie. Plus précisément, chaque calcul peut conduire à une déviation d'une unité ou d'une dizaine, et certaines formes particulières sont introduites dans des tables (exemple : $8 * 4 = 24$).

Le traitement complet de l'enquête sur les 4 exercices concernant la simplification a permis de vérifier la cohérence de cette approche. La plupart des résultats trouvés par les enfants sont interprétables. Les cas d'échec représentent un pourcentage négligeable (environ 1 résultat sur 6 correspondant à des erreurs rares). Ils sont dus principalement à des erreurs de calcul non intégrés dans la recherche (certains décalage de 2 unités par exemple), des erreurs de recopiage ou des cas qualifiés de monstrueux ou mystérieux dans la grille générale d'analyse (consulter en annexe V.2 les résultats complets associés à un exercice).

Au sujet des erreurs de calcul, on peut observer un phénomène assez étrange. Certains élèves, sur les fractions nécessitant un calcul préalable au numérateur, ont effectué une simplification abusive avant l'effectuation :

$$(51 + 26)/44 = (51 + 13)/22 = 64/22$$

Aucun des élèves ayant fait cette transformation n'a fait d'erreur de calcul. Une interprétation plausible consiste à penser que ces élèves ont une bonne maîtrise des opérations de base et qu'ils ont cherché une astuce inexistante dans l'exercice par une forme de déviation scolaire. Jugeant ce qui leur était demandé trop facile, ils ont essayé de découvrir quelque chose de plus subtil, i. e. de trouver une simplification camouflée. A contrario, on peut penser que les erreurs de calcul sont un indice de difficulté générale d'un élève face à un exercice, la concentration nécessaire pour résoudre une tâche concernant les fractions conduit à un relâchement dans les tâches de calcul sur les entiers (elles ne seraient pas dus à l'inattention mais à une forme de débordement).

V.2.b.2.b Persistance des erreurs

Une étude spécifique a été menée concernant les élèves ayant échoué aux 4 exercices sur la simplification, pour voir si les erreurs effectuées dans ces différents exercices pouvaient correspondre à des causes semblables. 129 élèves sont dans ce cas, et la moitié d'entre eux présente un comportement que l'on peut qualifier de systématique :

- recherche d'une valeur décimale de la fraction
- simplifier par 2 avant tout, si c'est impossible :
  - multiplier par 2,
  - ne rien faire
  - simplifier un seul des deux membres par 2
- trouver un seul facteur quand il n'a que cela à faire ou ne rien faire à l'issue de calculs intermédiaires,
- soustractions successives,
- simplification typographique.

On peut constater que beaucoup de ces erreurs sont liées à une incompréhension du but, i. e. de la simplification. Les simplifications hétérogènes ou d'autres calculs complexes ne semblent pas systématiques. Ceci renforce une hypothèse de type 'perte de sens' mais sans qu'il y ait d'algorithmes très persistants. On peut remarquer que ceci concerne principalement les élèves des petites classes (6ème, 5ème). Ainsi, la recherche systématique d'une division par 2 s'explique par le fait que les élèves n'ont pas de notion de diviseur premier ni celle de fraction irréductible et qu'ils essayent la seule chose qu'ils connaissent. On remarque d'ailleurs que des exemples présentés en 6ème ne montrant que des simplifications par 2 ou 5 sont susceptibles d'engendrer de mauvaises généralisations (REPAIR THEORY).

V.2.c Méthodologie de recherche de diagnostic

Dans le cadre d'une recherche sur la conception d'un environnement d'apprentissage de la résolution des problèmes de chimie quantitative prenant en compte les difficultés des élèves (voir [BLONDEL & Al. 90]), une étude a été menée avec BADAUD sur le diagnostic, à partir des premiers résultats de cette équipe sur l'analyse des erreurs [SCHWOB & BLONDEL 89]. Cette réalisation a permis d'enrichir la problématique initiale de BADAUD.

Remarque : ces deux recherches (fractions et chimie) font partie d'un cadre général sur l'utilisation des techniques d'intelligence artificielle pour la conception d'outils de formation. La mise en commun d'outils et de techniques intéressant des domaines de connaissance différents a pu être effectuée sur le diagnostic, et constitue un bon exemple de transdisciplinarité. Dans les deux cas, en ce qui concerne strictement BADAUD, le moteur est identique : l'aspect déclaratif est essentiel puisqu'il garantit une bonne indépendance entre la recherche du chemin et les divers opérateurs ou règles de transformation. Le contrôle à partir d'un ensemble de relations entre les erreurs s'effectue de manière identique. Les premiers résultats montrent la faisabilité de l'approche et sa généralité (dans le cadre des contraintes énoncées plus haut, V.2.a).

Faire un système de diagnostic est loin d'être avant tout un problème informatique. Il s'agit plutôt d'un problème didactique d'expertise sur les divers types d'erreurs faites par les élèves et leurs causes. Quand on se limite à rechercher des cheminements plausibles pour arriver au résultat trouvé par un élève, sans s'occuper de modéliser globalement le comportement de cet élève, comme c'est le cas dans BADAUD, les techniques informatiques nécessaires sont classiques (elles sont décrites précédemment).

Payne et Squibb (1988) (cité dans [HENNESSY 90]) mettent bien en évidence les conflits d'intérêt du diagnostic :

- pas de discrimination sur les erreurs individuelles pour construire les gros catalogues,
- discrimination à un niveau d'abstraction suffisant pour concevoir une théorie générative des erreurs.

BADAUD cherche à donner une réponse pragmatique à ce problème en dégageant une méthodologie basée sur la catégorisation des erreurs.

On distingue plusieurs étapes de réalisation :

1. **Recueil d'erreurs locales et de quelques comportements types**

   Cela correspond à la recherche de la nature des déviations élémentaires, ce qui nécessite une étude didactique préalable sur une population suffisante. Dans le cas des fractions et de la chimie, ces observations ont été menées, permettant d'obtenir un catalogue des erreurs effectivement produites. Ces études n'ont pas la prétention d'être complètes mais couvrent un champ très large, l'exhaustivité est ici impossible.

   Cette étape est nécessaire tant que l'on ne possède pas de théorie suffisamment fine de génération des erreurs permettant de les simuler d'une manière automatique (voir V.1.d, page 159). D'ailleurs, une telle théorie ne dispenserait pas d'une vérification expérimentale dans un domaine donné (cela supposerait de connaître de plus le cursus des élèves). L'aboutissement de cette recherche est la constitution d'une première typologie des erreurs faites par les élèves dans le domaine.

2. **Ecriture d'un résolveur perturbable**

   Cette construction d'un résolveur perturbable s'appuie sur la typologie précédente et prend en compte TOUTES les erreurs constatées. Il s'agit avant tout de choisir une représentation des connaissances du domaine adaptée au codage des bugs élémentaires trouvés. Une représentation déclarative semble la plus propice pour les mises à jour et corrections éventuelles, et un développement incrémental. On n'essaye pas dans un premier temps de contrôler l'explosion combinatoire, mais de s'assurer que l'on peut prendre en compte toutes les combinaisons d'erreurs imaginables. Des tests permettent de s'assurer que l'on retrouve bien les erreurs que l'on a introduites (et d'autres pistes peut être inattendues). Cette étape est une sorte d'étude de faisabilité et donne une idée précise de la complexité de la recherche (la grosseur de l'arbre de recherche fournit un indice sur le type de réductions qui s'avère nécessaire d'effectuer). On a une première ébauche du passage du local au global (cheminement global généré à partir de perturbations locales).

3. Test et validation du logiciel

Il s'agit tout d'abord de vérifier la prise en compte des erreurs détectées et d'analyser les phénomènes de dispersion et d'explosion (chemins dénués de sens, combinaisons à éviter, i. e. chemins différents aboutissant au même résultat par des inversions successives. On utilise une règle min-max : détecter le maximum d'erreurs en générant le minimum de chemins, éviter les interprétations multiples (leur associer un codage unique, exemple $42/63 = (6 * 7)/(9 * 7) = 6/7$ correspond à une simplification hétérogène ou au remplacement du quotient par le diviseur).

Ensuite, il faut déterminer des formes de contrôle à intégrer pour éviter les phénomènes d'explosion combinatoire sans perdre d'exhaustivité dans la recherche des chemins plausibles. Ceci se mène de pair avec une étude didactique reprenant la typologie initiale et la codant d'une manière voisine de celle du programme. Certaines réductions sont générales et indépendantes du domaine (elles correspondent en gros à celles introduites dans DPF [OHLSSON & LANGLEY 88]), d'autres réductions sont spécifiques du domaine considéré et permettent d'éliminer des chemins équivalents au niveau du diagnostic (en chimie, on supprime ainsi certains chemins mettant en cause les mêmes formules à des moments légèrement différents).

A partir de là, on dispose d'un outil capable de résoudre la plupart des cas d'erreurs d'élèves rencontrés et on peut affiner la typologie de ces erreurs.

4. Introduction d'un mode de contrôle de la recherche

Il faut trouver des formes de contrôle plus fines à partir d'un typage des erreurs et de relations entre elles. On tente d'intégrer au diagnostic comportemental une analyse des causes des erreurs rencontrées. On s'appuie sur une théorie locale des erreurs dans le domaine permettant de réduire l'écart entre le 'comment' et le 'pourquoi', i. e. associant des cheminements à des mauvaises conceptions générales des élèves. On introduit aussi des schémas de plans généraux utilisés par les élèves. Le contrôle doit pouvoir s'effectuer à partir des erreurs et de leurs relations, modélisant des comportements, i. e. un système de règles sur les types d'erreurs (ce qui suppose une étude de la stabilité).

Un bilan permet d'analyser ce qui peut être pris en compte dans le diagnostic, ce qui donne une idée des performances du logiciel (degré de liberté, notion à préciser qui devrait correspondre à un ensemble de paramètres indépendants pris en compte pour modéliser le comportement de l'élève).

On peut éventuellement introduire de nouveaux paramètres :

- type d'enseignement reçu (qui conditionne les erreurs produites),
- profils génériques (de type exécutable) fournissant des hypothèses sur les types d'erreurs susceptibles d'être rencontrées,
- contextes favorisant les erreurs (utiles pour la génération d'exercices).

Chapitre V.3

Diagnostic et enseignement

V.3.a Retour sur la notion d'erreur

V.3.a.1 Qu'est-ce qu'une erreur grave?

Les systèmes de diagnostic ont tout d'abord leur place en formation d'enseignants, qu'elle soit initiale ou continue. Il peut s'agir, dans un premier temps de les faire réfléchir sur la notion même d'erreur, dans ses manifestations et sa genèse, et sur leurs propres conceptions dans ce domaine.

Une idée reçue, que l'on trouve chez de nombreux enseignants, est celle d'erreur grave. Beaucoup d'entre eux, quand on leur représente certains calculs erronés, ont tendance à émettre des jugements de valeur tranchés en classant certaines erreurs comme des inattentions et d'autres comme des manifestations de lacunes graves voire d'incompréhensions totales. Leur faire prendre conscience que le cheminement observable (la trace) correspond à divers types d'erreurs profondes, que le même comportement peut s'interpréter de plusieurs façons, leur indique que cette notion de gravité est plutôt attachée à l'opinion que l'enseignant a de l'élève et que l'erreur n'est qu'un symptôme, manifestation de formes très diverses de 'maladies' (voir V.1.a.4, page 149). Ce type de diagnostic comportemental donne aussi un sens à ce que l'on qualifie trop souvent de 'n'importe quoi'. Ainsi, des calculs jugés totalement incohérents peuvent s'avérer répondre à une logique interne scrupuleusement suivie, et répondre à des buts ou correspondre à des croyances des élèves sur le domaine traité ou même sur les exercices scolaires en général (voir V.2.b.2, page 164).

La notion d'erreur grave est à relier directement à celle de complexité d'un problème (VI.4.c, page 202). Cette complexité recouvre en fait deux notions, celle de difficulté, référant aux efforts à déployer pour trouver un chemin vers la solution, et celle de taille ou de volume d'une solution. Un problème peut être facile, dans le sens où le chemin à emprunter est aisément repérable, et nécessiter des développements ou des calculs importants (c'est le cas de nombreux exercices de mathématiques scolaires centrés sur la maîtrise de manipulations formelles). Séparer ces deux formes de complexité est souvent un des objectifs des environnements d'apprentissages ou des tuteurs (éviter à l'utilisateur la conduite de calculs pour qu'il puisse se concentrer sur le choix des opérateurs, e.g. APLUSIX, MATHPERT, VI.2.b). A n'en

pas douter, le volume des calculs est un facteur important dans la production d'erreurs. Une forme d'encombrement mémoire dans l'application de procédures mal maîtrisées conduit à des débordements incontrôlés. C'est l'indice d'une maîtrise insuffisante, pas nécessairement le signe d'une incompréhension totale, quelque puisse être l'erreur (les conduites métacognitives de contrôle sont rarement mises en oeuvre). On constate une forme de relâchement des contraintes (non respect de règles pourtant connues), qui se traduit par des déviations non fixées, comme des erreurs de calcul.

V.3.a.2 Utilisation du programme BUGGY en formation

BUGGY [BURTON & BROWN 78] est un programme destiné à l'entraînement des enseignants dans le diagnostic de bugs en addition et en soustraction. Il se présente comme une sorte de jeu où l'utilisateur est censé être un expert consulté par une commission. Cet expert doit reconnaître, dans des opérations effectuées par des enfants, les erreurs systématiques faites. Au lieu de fournir un diagnostic sous forme verbale, l'expert doit montrer qu'il a reconnu l'erreur en donnant le résultat que donne l'élève sur cinq opérations consécutives. Il peut poser autant d'opérations qu'il désire à l'élève pour se forger une idée précise de son comportement.

Ce programme est intéressant à double titre :

- il sensibilise les enseignants sur ces problèmes de bugs et les familiarise avec les plus courants, ce qui les invite à en tenir compte dans leur comportement ultérieur face aux enfants (il ne s'agit pas de les entraîner à être des experts en diagnostic),
- le déroulement même du jeu invite à développer une démarche scientifique : exemples, conjecture, hypothèse, test, induction, vérification. En effet, un seul exemple ne suffit pas à assurer un diagnostic suffisamment sûr et conduit souvent à plusieurs interprétations. Il faut donc conjecturer et trouver des exemples à poser à l'élève pour infirmer ou confirmer ses hypothèses. Le choix de ces exemples est ainsi déterminant. Par exemple, si on prend le cas de l'enfant qui collecte toutes les retenues sur la colonne la plus à gauche, ceci ne peut se repérer que si on pose des opérations à 3 ou 4 chiffres. De même, dans les cas d'absence d'emprunt sur un zéro, il faut introduire des opérations où il est nécessaire d'emprunter sur un zéro.

BUGGY est utilisé à l'ENI Bonneuil depuis 1986, tant en formation initiale qu'en formation continue. Ce travail suivant un but direct d'apprentissage, il n'y a pas eu d'observation fine de menée. Cependant, les réflexions d'environ 200 stagiaires ayant passé un peu plus de 2 heures sur ce programme apportent quelques indications sur son intérêt. On a pu faire les constatations suivantes (voir aussi [DE CORTE et Al. 87]) sur une analyse du comportement des enseignants face au même logiciel) :

- les comportements face au logiciel sont très tranchés. Quelques uns ont une attitude de rejet, ils n'arrivent pas à trouver correctement les bugs et rencontrent des difficultés importantes à entrer dans cette activité. Ils reconnaissent néanmoins un intérêt à cette problématique. Les autres sont passionnés par cette recherche d'erreurs systématiques et essayent de résoudre le maximum de cas.
- les instituteurs en formation continue réussissent en général beaucoup mieux que les normaliens en formation initiale. De plus, dans un groupe d'enseignants du primaire

ayant des classes de différents niveaux, ceux ayant l'habitude des petites classes (CP, CE1) vont plus vite que ceux ayant la charge de grandes classes (CM1, CM2). D'ailleurs, un instituteur travaillant en SES (section d'éducation spécialisée) repérait les bugs avec une rapidité extrême. Ceci montre que les enseignants ont une connaissance pratique des diverses erreurs faites par les élèves, bien qu'ils n'aient pas dans leur classe de problématique de diagnostic fin (en fait, ce sont des connaissances peu formalisées et non institutionnalisés, i. e. n'ayant pas de statut professionnel).

- l'écart entre la méthode la plus utilisée en France (basée sur l'invariance de la distance, la retenue étant introduite sur le nombre du bas) et la méthode dite naturelle utilisée aux Etats Unis, mais présente encore en France (basée sur l'emprunt et plus proche de la numération de position) n'est pas un obstacle pour les enseignants ayant une pratique suffisante de la classe, mais handicape les personnes ne connaissant pas les différentes techniques de soustraction (c'est souvent une découverte pour les normaliens, qui n'ont que des réminiscences vagues de la technique de soustraction et maîtrisent de manière insuffisante les problèmes liés à la numération de position).
- les réactions sur un nouveau cas un peu plus subtil, les précédents ayant été résolus, font resurgir les a priori standards sur les erreurs faites par les élèves (« *celui est vraiment très atteint! C'est n'importe quoi! Il n'a rien compris! etc.* »). Cela offre ainsi une expérience concrète montrant que des comportements, trop rapidement qualifiés de farfelus et d'aléatoires, peuvent répondre à une logique interne dont la découverte peut donner des atouts pour effectuer une intervention efficace (ou tout au moins, éviter de faire une intervention totalement inefficace).
- une réflexion sur les techniques de soustraction et les difficultés associées permet de prévoir les erreurs qui risquent de se produire. On arrive ainsi à s'intéresser aux causes de ces erreurs (voir V.1.d, page 159). Les raisons sont assez simples (numération de position, problèmes d'échange) et permettent de situer les interventions à faire en tant qu'enseignant.

En conclusion, cet environnement d'entraînement au diagnostic des erreurs est très bien accueilli par l'ensemble des instituteurs, tant en formation initiale qu'en formation continue. Cette activité a d'ailleurs tendance à les surprendre, non parce que les problèmes de diagnostic leur sont inconnus (pour les enseignants confirmés), mais parce qu'ils ne se sont posé ce problème que dans le cadre strict de la remédiation.

V.3.b Traitement des erreurs

Diagnostiquer avec plus ou moins de précision les erreurs faites par les élèves est intéressant mais ne précise pas quel comportement adopter face aux erreurs :

- faire de la remédiation,
- essayer d'éviter leur émergence,
- les ignorer?

Ceci amène d'ailleurs de nouvelles questions sur les bugs : y a-t-il une stabilité des bugs le long du cursus, est-ce un frein pour de nouveaux apprentissages ou disparaissent-ils d'eux-mêmes?

V.3.b.1 Remédiation

La remédiation est un des problèmes les plus difficiles de la formation. Les études montrent qu'une action sur les connaissances erronées d'un apprenant est très complexe à mener et est loin de pouvoir avoir un effet garanti (voir par exemple le problèmes des étudiants et des professeurs, II.2.a.2, et les travaux menés en algèbre par D. Sleeman et son équipe [SLEEMAN & Al. 89]). De plus, si cette remédiation veut se baser sur le diagnostic, ce dernier doit être suffisamment précis et référer aux causes les plus profondes des erreurs détectées. On peut d'ailleurs voir là un paradoxe : les déviations superficielles non encore stabilisées sont faciles à réorienter surtout dans la phase initiale d'apprentissage (il n'y a pas alors obligatoirement de diagnostic précis), alors que les comportements liés à des erreurs systématiques peuvent être détectées de manière plus approfondie, mais sont extrêmement réfractaires aux tentatives de correction.

De Corte & Al. [DE CORTE et Al. 87] rapportent les études de Putnam [PUTNAM 85] qui montrent que les enseignants expérimentés n'essayent pas de construire des modèles très détaillés des procédures fausses des enfants avant d'effectuer des remédiations. En d'autres termes, ils poursuivent rarement le sous-but de déterminer la nature exacte de l'erreur d'un élève.

L. Resnick insiste, dans le cas de la soustraction, sur la recherche des causes profondes des erreurs, le diagnostic basé sur les déviations dans les algorithmes étant insuffisant [RESNICK 82] : " *Therefore it should be complemented by the development of diagnostic and remediatial teaching skills focusing at the concepts underlying the algorithmic procedures and at the multiple links between the syntactic and the semantic knowledge base.* " Les études suivantes montrent cependant que ce travail sur le sens de l'opération et les liaisons avec la numération de position (voir le paradigme des 2 mondes, II.2.a.3) n'éliminent pas ipso facto l'émergence de bugs.

V.3.b.2 Curriculum / Programmes

Si la remédiation s'avère problématique, on peut réfléchir sur le mode de génération des erreurs et essayer d'éviter leur émergence, ou tout au moins comprendre leur génération (voir les travaux de VanLehn : REPAIR [BROWN & VANLEHN 82], STEP, SIERRA [VANLEHN 87], CASCADE [VANLEHN 90]).

Malheureusement, s'il semble possible d'en limiter la production, leur émergence paraît presque indissoluble du fonctionnement scolaire traditionnel, ainsi que le note VanLehn (in [MAURER 87]) : " *many bugs seem to be generated by an interaction between spiraling in the curriculum on one hand and diagnostic and placement testing on the other.* "

Un des facteurs prépondérants en mathématiques est celui de la conservation ou de la perte du sens dans la manipulation des expressions. On retrouve l'opposition entre performance et compréhension (voir IV.3.e), forme et fond (IV.2.c.2, VI.3.b, page 194). Comme il a déjà été indiqué, un travail sur le sens ne permet pas d'éviter les déviations. Il donne cependant une plus grande aptitude aux enfants à expliciter les concepts liés à la soustraction (voir auto-debugging), ce qui est certainement un pas vers la réussite [CHI & Al. 89].

Une réflexion sur les programmes et les progressions scolaires peut cependant être bénéfique pour les enseignants :

- apporter un soin extrême à la succession des exemples présentés aux enfants et au découpage des difficultés;
- associer le plus possible forme et fond, i. e. travail sur le sens et sur les algorithmes;
- favoriser un apprentissage avec compréhension qui devrait donner des résultats transférables à divers domaines liés à la résolution de problèmes (voir [PERKINS & SIMMONS 88] qui conduisent une analyse sur les incompréhensions en science, mathématiques et programmation).

Les programmes scolaires officiels devraient prévoir les déviations quasiment inéluctables qu'ils peuvent engendrer : ajouter en annexe les modèles erronés que peuvent avoir les enfants pour en tenir compte les années suivantes (institutionnaliser les erreurs des élèves en les considérant comme des effets de bords de l'enseignement qu'il faut pouvoir diagnostiquer et surmonter). En particulier, les évaluations nationales menées en CE2 et en 6ème devraient dépasser la simple évaluation des performances pour dégager des profils cognitifs.

V.3.c BADAUD et EIAO

Les modules de diagnostic intégrés dans des programmes d'EIAO ont été vus précédemment (voir V.1.c, page 154). L'idée générale est de simplifier ou de restreindre le diagnostic en fonction des stratégies d'intervention ou de l'environnement proposé (forme plus ou moins prescriptive de l'outil).

L'intégration de modèles élèves trop pauvres dans les tuteurs intelligents, se concentrant sur des problèmes de surface, bute sur deux obstacles majeurs [LAURILLARD 90] :

- impossibilité de localiser les incompréhensions les plus profondes,
- perpétuation de la technicisation du processus concerné (au détriment d'un retour au sens).

Par contre, les modèles plus élaborés sont souvent difficiles à faire tourner 'en ligne' (voir par exemple [PALIES 88]).

BADAUD n'est pas destiné à être intégré dans un tuteur, mais est un outil générique de diagnostic qui concerne prioritairement les enseignants.

Les erreurs étant largement liées à l'enseignement dispensé, les catalogues et les procédures intégrées ne peuvent être complètes. Des mises à jour sont nécessaires et les évolutions liées aux changements de programmes et de méthodes d'enseignement conduisent à l'émergence de nouveaux types de déviation. On peut d'ailleurs noter qu'une étroite collaboration est souhaitable entre les enseignants et les chercheurs. Schoenfeld [SCHOENFELD 87a] donne ainsi deux exemples complémentaires d'une telle coopération. Dans le premier, un chercheur n'arrive pas à expliquer une anomalie dans ses données, une suite consistante de types d'erreur d'élèves qui diffèrent de ceux qu'il a déjà vus. Un professeur a pu lui indiquer que ce comportement pouvait être lié à une procédure précédente enseignée de manière non standard dans un livre. Réciproquement, un comportement d'élèves qui semblait incohérent à un enseignant a pu être expliqué par un chercheur, par une généralisation incorrecte d'une procédure de soustraction des entiers (transformer 7 24/40 en 6 124/40 avec une méthode d'emprunt).

L'idée est ici de considérer BADAUD comme une base initiale qui peut être complétée de manière incrémentale pour tenir compte de nouvelles erreurs.

On peut lui affecter trois types d'utilisation :

- un jeu à la BUGGY, mais ouvert, à partir d'une base d'erreurs la plus largement possible déclarative, Ce jeu pourrait d'ailleurs aussi être utilisé par les élèves, pour améliorer leur confiance en eux, interpréter leurs comportements fautifs, non comme des stupidités mais comme une source de données sur leurs propres erreurs. L'intérêt pour les enseignants a déjà été souligné (V.3.a.2, page 172), il devrait être le même dans d'autres domaines que la soustraction (il peut d'ailleurs sembler étonnant que ceci n'ait pas déjà été réalisé pour l'algèbre élémentaire, vu le volume important des recherches qui s'y sont consacrées).
- un outil de diagnostic automatique qui peut aider un enseignant en lui fournissant un certain nombre de chemins plausibles pour l'obtention d'un résultat. Cela pourrait fournir une aide concrète appréciable, vu les difficultés qu'ont les enseignants à cataloguer les erreurs des élèves. On peut imaginer aussi bien une génération a priori (avant de poser l'exercice), pour prévoir les résultats qui risquent d'émerger et permettre une intervention immédiate, ou a posteriori, comme des grilles de lecture possibles des productions des élèves.
- un outil pour le chercheur, en jouant sur les types d'erreurs et leurs relations pour déduire quelques profils généraux ou explorer les liens qui peuvent exister entre les diverses déviations. En effet, l'écart entre le codage manuel initial et les perturbations introduites dans le programme de diagnostic rend souvent ardu l'analyse globale des résultats d'une enquête. Les bilans statistiques n'offrent pas une grande souplesse et peuvent difficilement rendre compte des erreurs liées entre elles. Le système de diagnostic, contrôlé par des liens explicitement déclarés entre ces erreurs ouvre un champ d'expérimentation relativement large.

Sixième Partie

CAMELEON

Calcul Algébrique Mathématiques ELémentaires
(Etude d'une Organisation Nouvelle)
(Environnement Opérationnel Naturel ou Nouveau)

" ... a person's conceptualization of a problem may actually emerge as a by-product of the problem-solving strategy itself. Simultaneously, the way in which a problem is conceptualized will suggest a strategic ordering of problem-solving operations that can be applied to solve the problem. It is precisely this sort of reciprocal, dynamic interaction between representation and operation that must be supported if computers are to serve as conceptualizing tools in instruction. "

[DERRY 90]

L'idée générale est de construire un résolveur auto-explicatif, incluant une visualisation effective de son processus de recherche, ce dernier correspondant à celui d'un expert en résolution de problèmes en mathématiques, pas nécessairement expert dans des domaines plus restreints (par exemple dans le calcul des limites). L'accent est mis avant tout sur les connaissances déclaratives générales (connaissances procédurales uniquement en résolution). Cette approche s'éloigne des architectures classiques des systèmes experts et est fortement dépendante d'une structuration des connaissances mathématiques. Elle devrait permettre d'implanter des formes de tuteurs/assistants destinés plus à des interactions de collaboration avec l'apprenant. En particulier, le programme doit être capable de reconnaître les 'évidences' et d'exploiter les liens entre les connaissances.

Aucune implantation complète n'a encore été réalisée, mais de très nombreux essais ont montré la faisabilité de cette approche. L'annexe VI (voir A6) décrit un micro-modèle limité à l'étude de la parité et la résolution des équations.

Le lancement d'une telle implantation bute d'ailleurs sur le choix de l'outil adéquat pour la réaliser. On est confronté à une alternative :

- réécrire un système de calcul formel,
- coopérer avec un système de calcul formel (i. e. créer une sur-couche, d'une manière analogue à CAMELIA vis-à-vis de REDUCE).

La première approche permet un contrôle complet de toutes les formes de résolution mais oblige à un très gros travail d'écriture, pour un résultat problématique (résolveur trop lent, incapacité à traiter d'expressions complexes, etc.).

La deuxième approche oblige à préciser les protocoles de communication entre la couche pédagogique et le système de calcul formel. Cette forme de pilotage semble préférable, mais elle ne permet pas une implantation effective pour l'instant.

Ce chapitre se compose de trois parties, les deux premières devant permettre d'expliquer les choix effectués :

- chapitre 1 : historique,
- chapitre 2 : réflexion sur les environnements d'apprentissage liés au calcul formel et mise en évidence de certaines insuffisances,
- chapitre 3 : réflexion sur les objectifs pédagogiques que l'on peut assigner aux environnements d'apprentissage sur le calcul formel, réflexion sur la métacognition et l'enseignement des heuristiques,
- chapitre 4 : description de l'architecture de CAMELEON.

Chapitre VI.1

Historique

VI.1.a Les idées de départ

Le travail autour de CAMELEON était initialement un prolongement de CAMELIA en vue de réaliser un programme tuteur/assistant dans le domaine de l'étude des fonctions cartésiennes (niveau Terminale des Lycées). L'hypothèse de base n'était pas de faire un tuteur au sens strict, mais plutôt un assistant préfigurant la couche didactique d'un système de calcul formel. Cette perspective ouvrait un certain nombre d'interrogations :

1. **Interrogations d'ordre didactique :**
   - que faut-il enseigner sur ce sujet si les machines sont capables d'une résolution automatique?
   - que faut-il savoir pour piloter un système de calcul formel?
   - comment formaliser les méthodes de résolution et les liens entre les diverses connaissances mises en jeu?
2. **Interrogations d'ordre informatique :**
   - quelles contraintes donner à la résolution?

   Le risque avec les systèmes à base de règles est de suivre des cheminements encore trop éloignés d'un mode «naturel» de résolution. L'utilisation des formes fonctionnelles des théorèmes correspond à un mode a posteriori d'explications identique à celui trouvé dans les manuels scolaires, mode qui est souvent une reconstitution inversée d'un mode de recherche productif.

   En particulier, il faut apprendre à avoir le «coup d'oeil» : c'est une difficulté majeure qu'il faut enseigner (une forme d'expertise), et qui est liée à une démarche de nature ascendante (sur les expressions). On recherche tous les renseignements possibles sur l'expression à traiter, et toutes les formes que l'on peut lui donner (par des transformations élémentaires), ce qui conduit souvent à une résolution immédiate ou au choix motivé d'une méthode de résolution (voir VI.2.b.2, page 188).

   Le système doit non seulement savoir résoudre mais trouver la «meilleure» solution. Cette notion est floue tant que les critères d'optimisation ne sont pas explicités, et ne

peut se confondre avec la méthode la plus courte (on parle aussi de l'élégance d'une démonstration). Il faut aussi pouvoir suivre et découvrir plusieurs pistes. On peut d'ailleurs remarquer que l'échelle des difficultés n'est pas la même pour l'homme et la machine, même si on a pu constater parfois que les choses qui apparaissent difficiles à réaliser dans les logiciels coïncident fortement avec celles pour lesquelles les élèves ont des difficultés!

3. **Interrogations d'ordre cognitif :**
   - comment tenir compte d'un profil utilisateur?
     Un modèle d'élève succinct de type expertise partielle ('overlay model', voir V.1.b, page 152), correspondant à un petit nombre de descripteurs inclus dans les règles pour retirer ou favoriser des chemins suivant les connaissances supposées de l'élève, est-il suffisant?
   - comment organiser les connaissances?
     Il ne s'agit pas de mettre côte à côte des méthodes de résolution mais de représenter l'ensemble des connaissances nécessaires et de les rendre disponibles pour de multiples problèmes.

   En fait, on peut considérer que ce dernier problème est le plus fondamental et conditionne toutes les autres questions.

VI.1.b Les premiers travaux

A partir de l'explicitation de ces premières idées, le travail a débuté par la recherche de spécifications pour un résolveur, et le traitement d'exemples d'études de fonctions pour formaliser des modes de résolution. Pour des raisons diverses, essentiellement des mouvances au niveau des équipes et des institutions, la continuation du projet s'est effectuée sous des formes différentes. Les essais les plus concluants ont été effectués à l'aide du système SEVE (voir III.2.b.1, page 92) interfacé avec un système de réécriture développé en C par Paolo Machado (voir annexe A6).

Ces divers essais ont abouti à des réflexions fondamentales qui ont fait dévier peu à peu la problématique initiale :

- pertinence de l'outil : CAMELIA dans sa conception initiale s'est révélé insuffisant pour résoudre le but cherché. Les raisons sont développées plus loin (VI.2, page 185).
- pertinence du but : réaliser un tuteur/assistant apparût peu à peu comme une tâche trop complexe et finalement peu satisfaisante :
  - Est-ce un réel problème d'enseignement?
  - Que cherche-t-on à enseigner?
  - Comment l'enseigner?
  - Quelles sont les erreurs?

La finalité d'un tel système est sujette à caution : à vouloir être expert en tout, il risque d'être un simple répétiteur s'auto-justifiant par l'usage de quelques techniques locales.

Le stade actuel est centré sur la description d'un résolveur satisfaisant aux contraintes fortes exprimées en introduction. L'utilisation réelle en cadre scolaire nécessite d'autres réflexions

qui ne peuvent être occultées et qui ne sont qu'ébauchées dans la suite. La réalisation complète nécessite une équipe et la possibilité d'intégrer un système de calcul formel. Je souhaite que ce travail puisse se poursuivre dans cette optique.

Chapitre VI.1. Historique

Chapitre VI.2

Evaluation interne des résolveurs mathématiques

Dans ce chapitre, on reprend la perspective de Soloway et Littman (voir II.3.a.2) sur l'évaluation interne des ITS. On la limite ici essentiellement à la partie experte du domaine. Il s'agit de comprendre les différentes architectures implantées et les comportements associés. L'étude est centrée avant tout sur CAMELIA [VIVET 84], mais effectue quelques comparaisons avec d'autres systèmes (PRESS [BUNDY 79], APLUSIX [NICAUD 87], et d'autres). Dans ce cadre, les questions doivent être légèrement adaptées :

1. Que connaît l'ITS? De quelles connaissances disposent les divers résolveurs?
2. Comment l'ITS fait-il ce qu'il fait? Comment fonctionnent ces résolveurs?
3. Que devrait faire l'ITS? Quel environnement d'apprentissage ou quel tuteur bâtir?

Il s'agit d'ailleurs plus ici d'environnements d'apprentissages ou de résolveurs que de tuteurs intelligents. Pour distinguer les résolveurs utilisés dans le cadre de tuteurs des systèmes généraux de calcul algébrique, on désignera les premiers sous la dénomination de RPCA : résolveurs pédagogiques en calcul algébrique.

Remarque : deux caractéristiques principales de CAMELIA sont centrales :

- faire coopérer preuve et calcul,
- travailler comme surcouche d'un système de calcul formel (REDUCE).

Les développements prévus s'appuient sur ces deux aspects. Cette partie tente de recenser certaines limitations des différents résolveurs, limitations préjudiciables dans un cadre de formation :

- peu de connaissances sur les objets mathématiques,
- mode de contrôle limité,
- pas de reconnaissance explicite du type d'une expression,
- langage d'expression insuffisant.

VI.2.a Connaissances des RPCA

Les RPCA ont en général peu de connaissances sur les objets mathématiques de base eux-mêmes. L'accent est mis avant tout sur les méthodes de résolution. Les objets mathématiques interviennent principalement dans la phase de présélection des méthodes applicables. Leurs propriétés sont réduites à ce qui est directement relié aux règles de transformation (exemple entier pair pour la reconnaissance d'identités remarquables). Ainsi MATHPERT [BEESON 89] est dirigé avant tout par les opérateurs.

Ceci conduit souvent à un mode d'interaction où l'utilisateur modifie une expression en sélectionnant des transformations à l'aide de menus (voir [FERRET & JMENEZ 87] ou mode raisonnement et mode pilotage dans APLUSIX,.

CAMELIA dispose de connaissances plus étendues et traite quelques types particuliers : polynômes, fractions rationnelles, etc. En fait, les types d'objet interviennent dans les parties 'condition' des règles ou plans de résolution, ou dans les méta-règles stratégiques (PRESS).

Par contre, les connaissances de résolution sont représentées et utilisées de multiples façons : règles de réécriture (PRESS, CAMELIA), plans (CAMELIA), méthodes, opérateurs, méta-règles stratégiques ou de contrôle, etc.

Dans les systèmes qui se limitent à la factorisation de polynômes ou la résolution d'équations polynomiales (NAIADE, APLUSIX, etc.), les propriétés des objets mathématiques intervenant restent rares. Ce n'est plus vrai dès que l'on s'attaque à des problèmes plus généraux (fonctions transcendantes et fractions rationnelles, calcul de limites, etc.). CAMELIA dispose de connaissances sur les fonctions usuelles, mais elles sont réparties dans la base suivant les types de problèmes. Elles sont données telles quelles, non déduites à partir de propriétés générales, ce sont des résultats connus bruts qui ne peuvent donc pas être explicités.

`PRIM (E**X,X) --> E**X` (la primitive de e^x est e^x) est une règle présente dans le chapitre INTEGRATION. Mais, il n'est pas possible de retrouver l'ensemble des propriétés standards de la fonction exponentielle (croissante, sa propre dérivée, etc.). On accède à une connaissance uniquement par l'intermédiaire d'un problème.

CAMELIA dispose néanmoins d'un langage mathématique déjà relativement développé, mais il est réservé à l'écriture interne des plans. Une extension de ce langage est primordiale.

VI.2.b Fonctionnement des RPCA

Les RPCA adoptent le type de fonctionnement des systèmes experts. Ils diffèrent les uns des autres dans certains aspects de la stratégie de contrôle et dans leur utilisation de méta-règles. L'idée générale est de tenter d'unifier une forme de problème ('pattern') et des règles ou des plans généraux de résolution.

On retrouve trois étapes classiques (en employant ici le terme de transformations qui peut signifier aussi bien plan (CAMELIA), que règle de réécriture (PRESS), règle de production, méthode, etc.) :

1. Déterminer l'ensemble des transformations applicables au problème courant.

   Ceci correspond à une présélection qui peut inclure l'unification (APLUSIX), ou se limiter à des caractéristiques plus grossières (opérateurs présents dans l'expression comme

dans CAMELIA). Cette phase est souvent considérée comme automatique ou immédiate, bien qu'elle nécessite une expertise certaine (un 'coup d'oeil'). Le concept de 'visibilité' introduit dans APLUSIX permet de rendre compte de cette difficulté, mais il semble utilisé avant tout pour l'explication des actions.

2. Sélectionner la transformation qui semble être la plus intéressante.

   Ce choix s'opère grâce à l'application de méta-règles qui travaillent sur le problème courant (ou l'agenda des problèmes), et l'ensemble des transformations pré-sélectionnées. CAMELIA effectue un classement à partir d'un calcul coût/espoir [VIVET 83], APLUSIX qualifie les transformations envisagées (t-faible, faible, médiocre, moyen, a-bien, bien, t-bien).

3. Appliquer la transformation.

   Elle s'applique au problème courant dans le cas de CAMELIA. APLUSIX garde un agenda de problèmes et peut décider de revenir à un problème jugé plus intéressant.

Il faut remarquer que CAMELIA traite de problèmes difficiles (intégration, limites, etc.) et de natures diverses. En particulier, un problème d'intégration peut nécessiter la résolution d'un sous-problème de parité (condition-problème dans un plan). APLUSIX se limite pour l'instant à la factorisation et la résolution d'équations polynomiales, les problèmes sont de même nature et peuvent donc être comparés directement. Néanmoins, l'absence d'un agenda de tâches et d'une base de faits sont des limitations importantes de l'implantation actuelle de CAMELIA. La méthode utilisée dans PRESS est légèrement différente (voir plus loin).

Les trois exemples qui suivent sont là pour illustrer ce mode de résolution, et voir les divers problèmes qu'il peut poser. Le premier exemple : étude de l'intégrale d'Edimbourg vise à souligner les difficultés classiques de constitution de bases de connaissances (complétude, cohérence, fiabilité, stabilité). Le deuxième exemple montre qu'on peut opposer une analyse ascendante à la conception descendante habituelle. Le troisième exemple indique divers problèmes proches bien que non numériques.

VI.2.b.1 L'intégrale d'Edimbourg

Trouver $I = (d/a)^2 \int z^2.g((z-b)/d)dz$
où $g(x) = \arcsin x - x\sqrt{(1-x^2)}$ et $d = a.tg\ a$

Ce calcul d'intégrale est intéressant puisqu'il résiste aux techniques utilisées dans les logiciels de calcul formel (ex. REDUCE et MACSYMA).

L'astuce de résolution consiste à intégrer par parties avec $w = (z-b)/d$:

$$\begin{array}{ll} dv = (wd+b)^2 & v = (1/3d)(wd+b)^3 \\ u = g(w) & du = 2w^2/\sqrt{(1-w^2)} \end{array}$$

En fait, la dérivée g' est plus simple que g (suppression de la fonction Arcsinus), ce qui autorise l'intégration par parties en augmentant le degré du polynôme. Un changement de variable (w en $\cos x$) permet de terminer le calcul.

La comparaison entre une résolution manuelle et les plans de Camelia montre que ce dernier possède toutes les connaissances nécessaires pour trouver la solution. Un pilotage en pas-à-pas du système aboutit sans difficulté à la solution. Une étude détaillée a été faite au Centre

Mondial de l'Informatique sur l'implantation VAX de CAMELIA réalisée par Pierre Touzeau. Ce problème a d'ailleurs été résolu récemment avec CAMELIA [VOUILLE 90].

La trace des essais successifs de CAMELIA montre que les points clés du raisonnement (intégration par parties en augmentant le degré du polynôme, changement de variable en cosinus) sont classés dans la liste des plans possibles, mais en position non éligible. Des modifications minimes permettent d'éliminer cet obstacle (filtrage initial plus fin avec sélection par opérateur principal, évaluation de choses 'évidentes'). Néanmoins, le choix qui aboutit nécessite de nombreux essais et le développement d'un grand nombre de pistes infructueuses. Le classement des plans candidats s'effectuant à partir d'un calcul COUT/ESPOIR, il est possible de favoriser un plan en lui attribuant de meilleures notes. Cependant, ceci peut avoir des effets de bord difficilement contrôlables dans d'autres types de contexte. Une solution consiste à caractériser de manière suffisamment précise ce contexte pour que cela ne perturbe pas d'autres résolutions. Malheureusement, sans théorie solide permettant d'associer les types de problèmes aux méthodes de résolution, il subsiste une grande part d'improvisation.

Les techniques d'apprentissage automatique (voir LEX [MITCHELL & Al. 83] ou LP [SILVER 84]) ne semblent pas encore assez mûres pour apporter une réponse entièrement satisfaisante. On risque ainsi de se retrouver avec un résolveur censé pouvoir guider l'élève et parfois moins performant que lui (voir INTEGRATE [KIMBALL 82]).

Cette expérience illustre bien certaines difficultés de développement de résolveurs à base de règles, si on désire enseigner à l'élève des heuristiques dont on ne maîtrise pas complètement l'application (voir VI.3.b, page 194).

VI.2.b.2 Une étude sur la parité

Dans tous les exemples suivants, il faut étudier la parité de la fonction proposée. Il faut donc soit prouver qu'elle est paire, soit qu'elle est impaire, soit qu'elle n'est ni paire ni impaire.

L'idée est de montrer ici que, bien qu'une méthodologie utilisant des plans comme CAMELIA est susceptible de trouver une solution, le mode de recherche et de contrôle utilisé s'en éloigne sensiblement. En particulier, le fonctionnement analytique à la CAMELIA ne reflète pas le mode de résolution de l'expert, et des explications plus profondes nécessitent un traitement important de la trace et une reconstruction complète du raisonnement.

Certaines améliorations de CAMELIA permettraient de prendre en compte la plupart des difficultés soulevées, mais ne répondraient pas aux objections de fond (connaissances des objets, mode de contrôle).

1. $f(x) = \sin x + tg\ x$

   On montre que la fonction est impaire comme somme de deux fonctions impaires connues. La démarche descendante (utilisée par CAMELIA) s'appuie directement sur ce théorème (la somme de 2 fonctions impaires est une fonction impaire), l'opérateur principal de l'expression étant «+», il suffit de montrer que ses deux arguments sont des fonctions impaires impaires, ce qui est réalisé en consultant la table des résultats connus. On peut remarquer que, sur ce premier exemple, un «expert» a une réponse quasi immédiate. En fait, une manière plus naturelle d'opérer consiste à regarder les fonctions dominant les deux occurrences de x, i. e. sin et tg, de remarquer qu'elles sont

impaires et qu'il n'y a plus qu'à recoller avec l'opérateur «+». C'est une démarche ascendante (dans l'arbre de l'expression), on collecte des renseignements sur l'expression qui s'identifie alors à un théorème connu. On ne peut parler de véritable recherche. Le travail sur l'expression est préalable à toute sélection de plans de résolution.

2. $g(x) = \cos x * (x + 1) * (x - 1)$

Sur ce deuxième exemple, on peut tout d'abord remarquer que $g(1) = g(-1) = 0$. Ainsi, il faut choisir d'autres valeurs pour savoir s'il faut montrer que cette fonction est paire ou impaire. Le fait que $g(0) = -1$ suffit pour décider (g étant continue, si g était impaire, on aurait $g(0) = 0$). Ensuite, une résolution à la CAMELIA pose des problèmes délicats. En effet, le plan logique consiste à considérer g comme un produit et à chercher à montrer que tous les arguments de ce produit sont pairs ou impairs. La première étape réussit puisque cos est une fonction paire, mais le deuxième argument se montre réticent : $(x+1)$ n'est ni pair ni impair. Cet échec est rédhibitoire pour CAMELIA qui essaie alors un nouveau plan pour g. Or, l'échec de cette piste apporte une information importante non utilisée qui est que g est paire si et seulement si $(x + 1) * (x - 1)$ est paire. Ce dernier problème est d'ailleurs facilement résolu (c'est un polynôme). En conservant un système avec des plans indépendants, résoudre ce genre de difficulté amène une gestion plus complexe nécessitant des effets de bord au cours de la recherche. Une étude ascendante résout ce problème de manière naturelle : les facteurs pairs sont repérés directement (ici simplement $\cos x$), ce qui conduit à réduire la recherche à une expression polynomiale. De plus, ceci ne dépend pas de l'ordre des facteurs (on aurait pu exprimer g sous la forme $g(x) = (x+1) * \cos x * (x-1)$) qui aurait fait échouer le premier plan de CAMELIA directement.

Remarque : cet exemple est résolu par CAMELIA, à partir de la définition même de la parité. Cela pose néanmoins certains problèmes de simplification que CAMELIA ne prend pas en charge puisque gérés directement par REDUCE.

3. $h(x) = \cos \log \sin tg\ (1 + x^2)$

Cette fonction biscornue est pourtant reconnue comme étant paire directement. En effet, x apparaît une seule fois sous la forme x^2 ce qui permet de conclure. L'étude descendante est par contre plus que laborieuse. On peut aussi remarquer que si on est seulement intéressé par le fait de montrer que h est paire, il vaut mieux éviter de rechercher son ensemble de définition. De même une fonction telle que $h1(x) = (x^2+1)/(x^2-1)$ contient deux occurrences de x sous la forme x^2, ce qui permet de montrer directement qu'elle est paire et peut suggérer des changements de variables pour d'autres types de problèmes. Cette démarche est bien sûr explicitable sous forme de plans, seul le mode de contrôle peut éventuellement être problématique.

4. $i(x) = \cos(\pi/2 - x)$

Là encore, l'application des théorèmes peut suggérer une piste vouée à l'échec. cos étant pair, on peut chercher à montrer que son argument est pair ou impair. L'étude de l'expression dominant $x, (\pi/2 - x)$ n'étant ni paire ni impaire, cela montre qu'il faut au contraire simplifier ou transformer l'expression, et l'égalité $i(x) = sinx$ résout trivialement le problème. Ceci bien sûr est aussi dû au fait que cos est une fonction périodique.

5. $j(x) = Log\ ((x+1)/(x-1))$

   C'est le premier exemple non trivial. Les propriétés de la fonction log permettent de simplifier sensiblement le calcul. En effet, comme $\log(1/x) = -\log x$ pour montrer que $j(x)$ est impaire, il faut et il suffit de montrer que $k(-x) = 1/k(x)$ avec $k(x) = (x+1)/(x-1)$.

6. $\exp g(x)$ ou $\sqrt{g(x)}$

   De telles fonctions ne peuvent être impaires puisqu'elles ne prennent que des valeurs positives. Elles sont paires si et seulement si leur argument est pair.

7. $l(x) = E(x + 1/2)$
   où $E(x)$ désigne la partie entière de x

   C'est un piège du fait que pour toute valeur entière $l(n) = -l(-n)$. On utilise une méthode spécifique pour déterminer un contre-exemple (choisir x de telle sorte que l'argument de E soit entier). Ainsi, $l(1/2) = 1$ et $l(-1/2) = 0$. Il faut remarquer que ce type de choix est lié à la fonction partie entière, pas au problème particulier de la parité.

8. $m(x) = (\exp x - 1)/(\exp x + 1)$

   Dans ce dernier exemple, on remplace x par $-x$ dans l'expression et on cherche à la comparer à m.

$$\begin{array}{rcccl} m(-x) & = & (\exp -x - 1)/(\exp -x + 1) & = & (1/\exp x - 1)/(1/\exp x + 1) \\ & = & (1 - \exp x)/(1 + \exp x) & = & -m(x) \end{array}$$

   On remarque ici que pour trouver $m(1) = -m(-1)$ il faut faire un calcul similaire.

9. $n(x) = tg\ x$

   Si on ne veut pas se satisfaire de la simple mention résultat connu, il est nécessaire de pouvoir remplacer tgx par $(\sin x)/(\cos x)$, et donc le quotient d'une fonction impaire par une fonction paire. Ceci suppose simplement de pouvoir accéder à des expressions différentes de la même fonction, et, comme dans l'exemple (4), pouvoir transformer l'expression dans une forme plus adaptée au problème courant.

VI.2.b.3 Problèmes plus généraux

Les exemples suivants sont très simples et montrent que d'autres problèmes, qui dépassent le cadre des systèmes classiques de calcul formel, peuvent être résolus en étendant les capacités de CAMELIA. Les connaissances de résolution sont d'ailleurs très souvent déjà disponibles, mais le langage d'expression est insuffisant pour traiter de tels problèmes.

1. Etudier la parité de $f(x) = x^n$

   Ce n'est pas un résultat connu, il faut pouvoir raisonner par récurrence. Le résultat serait immédiat, si on pouvait exprimer des conditions sur un paramètre.

2. Montrer que le produit de deux fonctions impaires est une fonction paire.

   De même, CAMELIA peut résoudre ce problème, mais on ne peut pas lui poser! (cela revient à déclarer un objet non par une expression, mais comme élément d'une classe et vérifiant des propriétés particulières)

3. Soit f telle que $def\ f = R+$, étudier la parité de f

   Ici, une connaissance sur une propriété d'une expression permet de résoudre directement. Ce type de problème nécessite une base de faits pour stocker tous les résultats trouvés et les liens entre les divers problèmes.

4. Trouver une fonction f à la fois paire et impaire :

   Sans difficulté, la condition impose que f soit la fonction nulle. Pour tout x, $f(-x) = -f(x) = f(x) \Rightarrow 2f(x) = 0 \Rightarrow f(x) = 0$ Le raisonnement formel est immédiat, il nécessite uniquement de pouvoir traiter des fonctions comme un objet.

VI.2.c Cadres d'utilisation en formation

Il ne peut pas y avoir de réponse unique à la question de l'utilisation des divers types de systèmes de calcul formel dans un cadre de formation. On peut tout d'abord se servir des outils en tant que tels, comme des super-calculettes, suivre une approche de type micromonde, ou les utiliser dans une interaction tutorielle. On retrouve néanmoins d'une façon ou d'une autre un apprentissage par la résolution de problèmes. Il faut noter que ces possibilités de résolution automatique devraient avoir une incidence sur les programmes mathématiques eux-mêmes. On peut se poser la question de savoir s'il est plus important de maîtriser les techniques de calcul que de savoir utiliser efficacement des résolveurs (et, réciproquement, peut-on utiliser efficacement un résolveur sans maîtriser les techniques de calcul?).

On a ici deux approches différents :

1. travailler avec des systèmes de calcul formel à destination non prioritairement pédagogique (REDUCE, MUMATH, DERIVE, etc.). L'adaptation nécessite souvent l'intégration de systèmes d'aide ou d'assistance comme MACSYMA ADVISOR [GENESERETH 82], voire des systèmes plus explicatifs.
2. travailler avec des architectures bâties autour de RPCA, des couches didactiques autorisant diverses types d'exploitation (voir NAIADE, APLUSIX). Dans ce cas, on se trouve le plus souvent dans une problématique d'acquisition d'expertise, les interactions permettant de se décharger d'une partie du travail sur la machine et se concentrer sur des aspects plus stratégiques ('conceptual fluency tools' [PEA 87]). Les explications fournies par le système sont alors déterminantes. On retrouve ici la notion de fidélité cognitive (voir II.2.a.1, page 49). Elle intervient dans les trois étapes classiques d'un pas élémentaire de résolution :
   - présélection,
   - choix (résolution des conflits),
   - application.

Les explications sont a posteriori et si elles peuvent justifier un choix elles s'appuient sur les connaissances disponibles dans le système.

Les techniques de type hypertexte, surtout pour les formes d'accès à la connaissance peuvent résoudre une partie des problèmes. L'idée est de pouvoir fournir une trace à peu près adaptée et de permettre à l'élève de l'explorer lui-même suivant ses propres besoins, partant du principe qu'il sait mieux que le système les obstacles qu'il rencontre (ce principe est cependant loin d'être universel, un guidage pouvant souvent s'avérer nécessaire). Des liens avec les objets mathématiques manipulés peuvent aussi être fournis, ce qui autorise un accès aux diverses connaissances mises en jeu (voir Hyper-Arria, IV.3).

Chapitre VI.3

Metacognition et enseignement

" It seems clear that no process model of problem solving in any domain can be complete without an adequate account of the role of metacognition and belief systems. " [SILVER 87]

VI.3.a Métacognition et résolveurs mathématiques

Les philosophies de l'éducation reconnaissent que le contenu de la connaissance est beaucoup moins important que les aptitudes de plus haut niveau, telles que savoir raisonner sur la connaissance, acquérir de nouvelles connaissances, les adapter, etc. L'un des objectifs de l'éducation, dans une perspective de formation permanente, est de donner aux apprenants les capacités d'acquérir de nouveaux savoirs pour faire face aux changements quasiment inévitables dus aux nécessités professionnelles. Ceci amène tout naturellement à s'intéresser aux conduites métacognitives, c'est-à-dire aux conduites manifestées durant la réalisation d'une tâche en vue d'utiliser les ressources dont on dispose avec un certain degré d'efficacité [Flavell 79]. Les études en mathématiques montrent que les élèves n'ont pas d'habileté métacognitives, i. e. aucune connaissance ni contrôle sur leurs propres processus cognitifs. On peut d'ailleurs faire un parallèle avec les capacités d'auto-debugging des activités de programmation qui sont aussi peu développées [CARVER 87]. Schoenfeld [SCHOENFELD 87] fait une comparaison entre les temps consacrés par les élèves et les experts dans la résolution d'un problème, en décomposant les diverses phases : lire, analyser, explorer, planifier, implémenter, vérifier. Il note que le mathématicien passe une grande partie de son temps à 'réfléchir' plutôt que d'agir. D'une façon, plus précise, il passe beaucoup de temps à analyser le problème. Dès qu'il est parvenu à le maîtriser, travailler les détails de la solution n'est pas très long. Ceci rejoint les recommandations de Polya pour résoudre un problème [POLYA 57]. Il distingue 4 étapes principales : comprendre le problème, faire un plan, suivre le plan, regarder en arrière. Il est aussi primordial de trouver des représentations multiples du problème pour faire émerger la solution.

La notion d'évidence en mathématiques est ainsi très particulière. Quand on fait résoudre différents types de problèmes, nécessitant peu de connaissances mathématiques, à des futurs instituteurs, ils ont tendance à classer ces problèmes en deux catégories :

- ceux qui sont «évidents», i. e. ceux qu'ils ont réussi,

- ceux qui sont très difficiles, i. e. ceux qu'ils n'arrivent pas résoudre.

En fait, ils considèrent très souvent que l'activité consistant à trouver une représentation adéquate ne fait pas partie de la résolution, et comme il y a peu de calculs à effectuer, on tombe sur cette forme d'idée reçue de 'tout ou rien'. On note aussi l'extrême importance des croyances des élèves sur leur capacité à faire des mathématiques. Une de celle-ci, qui consiste à croire que tout problème peut se résoudre en 10 minutes ou moins conduit souvent à interrompre la recherche au bout de quelques minutes, même s'il est possible d'arriver au bout avec un peu d'effort (voir preuve-construction, IV.1.a.1, page 113)

L'une des conséquences de ces remarques, au plan de l'enseignement est de savoir s'il est possible de développer des aptitudes au niveau de la métacognition, et comment construire un environnement susceptible de favoriser ce développement (Voir [DILLENBOURG 90]).

VI.3.b Enseignement des heuristiques

Dans le cadre des RPCA (voir VI.2, page 185), on distingue trois étapes dans la résolution :

1. Détection des méthodes applicables au problème courant,
2. Choix de l'une de ces méthodes,
3. Application de la méthode, obtention d'un nouveau problème.

Cette vision locale doit bien sûr être complétée par des objectifs plus stratégiques contrôlant globalement l'ensemble de la résolution. Les RPCA tendent maintenant à accorder une importance prépondérante aux explications et à favoriser un enseignement des heuristiques. Cette position amène un certain nombre de réflexions.

Tout d'abord, il faut distinguer deux types d'heuristiques :

- celles de contrôle, correspondant à des aspects métacognitifs et qui sont indispensables à un niveau global;
- celles de choix de méthode correspondant à la phase de résolution des conflits dans un système expert pour éviter les recherches vouées à l'échec et des phénomènes d'explosion combinatoire. En fait, il s'agit essentiellement des connaissance liées au domaine précisant les méthodes prometteuses dans des cas prototypiques.

L'idée d'enseigner ces dernières heuristiques répond à un problème classique dans l'enseignement. Trop souvent, le cours de mathématiques introduit des concepts sous une forme déclarative, des exercices d'applications amenant des équivalents procéduraux nécessaires à la résolution. Très souvent, ces exercices permettent l'application directe des techniques enseignées. Le problème de choix de la technique ne se pose pas. Dans les exercices récapitulatifs, par contre, différentes méthodes vues peuvent être applicables, le choix de la (ou les) bonne méthode devient crucial. De nombreux élèves réussissent les exercices d'application et échouent dans les exercices récapitulatifs, ne sachant pas comment y associer les bonnes techniques : on constate une bonne performance quand la technique est proposée mais une absence de critère de sélection dans les cas généraux. Pour répondre à cet écueil, un travail spécifique sur le choix des techniques semble indispensable.

Ceci pose cependant un certain nombre de questions.

Tout d'abord, ce type d'heuristiques n'est pas très éloigné de ce que l'on appelle communément des recettes : on associe un 'pattern' à une méthode selon des critères plus ou moins obscurs. Enseigner ces heuristiques peut être un danger si elles sont trop locales. Elles peuvent cacher un manque de connaissances profondes du domaine et bloquer l'émergence de niveaux supérieurs d'abstraction. En effet, comment généraliser efficacement ce qui est vu dans une situation particulière, quels sont les critères amenant des inductions non abusives. On retrouve d'ailleurs les déviations attachées à la surgénéralisation. Si le choix des méthodes s'effectue à partir de reconnaissances de surface, des similarités non pertinentes peuvent conduire à la systématisation de mauvaises sélections. Maurer [MAURER 87] signale ainsi que pour toutes les procédures 'buggées' observées jusqu'à maintenant, les généralisations sont basées sur des propriétés de surface plutôt que sur le sens (voir les travaux de Vanlehn et Rissland)

La qualité des heuristiques est ainsi primordiale : il faut des connaissances approfondies et une grande pratique du domaine pour pouvoir les formuler et savoir correctement les appliquer. C'est une problématique de transfert de connaissances entre un débutant et un expert. La question sous-jacente est de savoir si l'objectif est de former des experts dans le domaine considéré (voir IV.3.e, page 143).

On retrouve ici le problème d'opposition entre le fond et la forme (la sémantique et la syntaxe) : Est-ce que les élèves peuvent faire des sélections stratégiques d'opérateurs sans une compréhension détaillée de la manière d'appliquer les opérateurs (sens des opérations vs techniques). Les études de Singley [SINGLEY 88] montrent que la maîtrise des opérations est un préalable. Un transfert asymétrique a été observé entre la sélection d'opérateurs et l'application d'opérateurs, dans le fait que l'application s'est transférée sur la sélection, et pas vice versa. De plus, les élèves qui faisaient les sélections en laissant le tuteur leur donner le résultat des calculs, avaient tendance à faire plus d'erreurs que les élèves qui conduisaient eux-mêmes les calculs.

Ainsi, un système comme ALGEBRALAND (Foss [COLLINS & BROWN 86]), libère les élèves du fardeau du calcul effectif avec les opérateurs choisis, pour qu'ils puissent se consacrer à des aspects stratégiques et de plus haut niveau conceptuel de la résolution de problèmes (forme d'interaction assez classique en mathématiques). Cependant, certains points critiques de vision stratégique dans la résolution de problèmes ne sont possibles qu'après que les étudiants aient automatisé des routines de bas niveau d'application d'opérateur.

Un certain nombre d'heuristiques, que l'on pourrait qualifier de scolaires, donnent de bons résultats :
Par exemple, l'axiome de Saint-Cyr cité par [LACOMBE 88b] :
« Tout lieu géométrique est un cercle ou une droite »
Démonstration : ce sont les seules courbes au programme.
De même, un axiome très utile en Terminale :
« Tout polynôme de degré supérieur ou égal à 3 a pour racine 0, 1 ou -1. »
(qui admet une forme procédurale très simple : essayer 0, 1 ou -1)

Ici, ce n'est pas la connaissance du domaine qui permet de les formuler, mais plutôt les contraintes du système éducatif et la connaissance des exercices scolaires. On arrive à une conception des règles excluant tout sens, celles-ci étant d'une nature qui oscille entre le métaphysique et le juridique. *« Mais elles forment rarement un système suffisamment impératif pour qu'il n'y ait pas le choix entre différentes possibilités, c'est-à-dire des heuristiques exactement comme un avocat malin peut conseiller à son client diverses stratégies juridiques pour*

légalement ne pas payer ses impôts. »[LACOMBE 88b] (métaphore juridique éloquente!).

Cette optique d'enseignement des heuristiques touche des domaines où elles sont indispensables, puisqu'il n'y a pas d'algorithme général garantissant l'obtention d'une solution (calcul de primitives, résolution d'équations, problèmes de géométrie, etc.). C'est d'ailleurs dans ces domaines que des recherches en IA sur l'apprentissage automatique (intégration LEX, équations LP,...) sont menées. On aboutit à une problématique de contrôle des heuristiques dégagées, problème qui est extrêmement complexe (voir VI.1.b.1, problème de stabilité).

Cette problématique d'heuristiques est bien une problématique d'expertise, nécessitant un temps forcément long d'apprentissage. L'étude de règles trop spécifiques peut être contre-indiqué. Celles utilisées dans SEDAF [AIELLO & Al. 88] sur l'étude du domaine de définition, par exemple, se limitant aux fractions rationnelles, ne concernent que les zéros du dénominateur. Cette particularisation qui semble excessive amène une vision pour le moins restrictive du problème. Par ailleurs, les critères de choix se basent le plus souvent sur des critères de dissemblance (sauf ce qui concerne l'analogie) ce qui s'oppose à la recherche de ressemblances et d'invariants, caractéristique des concepts mathématiques.

Par ailleurs, l'absence de procédures de contrôle est fortement préjudiciable, et est indépendante des heuristiques de sélection. On arrive à des stratégies comparables à celles des élèves besogneux qui n'exercent aucun contrôle sur leur travail, pas de réflexion sur l'exploration en cours, avec un risque important de se perdre dans les calculs longs et fastidieux! Le problème du choix initial peut être amélioré, mais les seules caractéristiques connues (statiques) ne permettent pas de faire le choix optimal (impossible de faire du reroutage dans les plans, trop de mélange de types de connaissance). Schoenfeld [SCHOENFELD 87], dans ses études, souligne que les élèves sont remarquablement consistants et persistants dans la poursuite de voies sans issue.

Pour résumer, les heuristiques de sélections et les processus de contrôle sont de natures différentes :

- les premières sont complètement dépendantes du domaine, et leur apprentissage est lié à une acquisition d'expertise, ce qui est peu transférable à d'autres domaines.
- les seconds sont très généraux et peu dépendants des domaines d'application, donc susceptibles de se transférer à différents domaines. (les deux types d'heuristiques sont cependant nécessaires, dans un processus de nature dialectique, voir IV.3.e)

Par exemple, l'idée consistant à essayer de se débarrasser des fonctions désagréables ('nasty' dans PRESS) est très générale, mais la notion de 'désagréable' est relative au problème que l'on cherche à résoudre. Disposer de la connaissance générale, et savoir l'instancier dans un domaine est plus puissant que la technique locale consistant à éliminer telle fonction précise pour résoudre un type de problème spécifique (voir AM [DAVIS & LENAT 82] pour un éclairage différent).

VI.3.c Un expert en résolution

Schoenfeld [SCHOENFELD 87b] explique comment, dans son cours sur la résolution de problèmes, il est passé d'un travail sur les heuristiques (développant la compétence dans divers domaines) à une tentative de développement d'une culture mathématique, i. e. des

heuristiques locales, à un changement global de comportement face aux mathématiques. Sa méthode s'articule autour de trois questions qu'il pose aux étudiants durant leur recherche de solution :

1. Que font-ils?
2. Pourquoi le font-ils?
3. Comment la réussite de ce qu'ils sont en train de faire va-t-il les aider à trouver une solution au problème?

Un système, expert en résolution de problèmes dans un domaine, devrait être capable de répondre à ces questions durant sa recherche!

Les considérations précédentes permettent de proposer certaines caractéristiques à une nouvelle implantation de CAMELIA ayant pour objectif de favoriser la mise en oeuvre de conduites métacognitives. Il s'agit de concevoir un système de type collaborateur, i. e. pouvant être utilisé comme outil ou prenant en charge un guidage local :

1. Présentant des résolutions plutôt que des solutions. C'est la recherche qui est importante, non la trace de la solution.
2. Fidélité psychologique de la recherche : tous les choix et les essais devraient être explicables (3 questions de Schoenfeld), de même que les modes de contrôle.
3. Système auto-explicatif : c'est son comportement qui tient lieu d'explication sous une forme largement non langagière. Ceci suppose de pouvoir représenter le processus de recherche de manière interne et externe.
4. Rôle central des représentations : savoir gérer plusieurs représentations et passer de l'une à l'autre en fonction du problème.
5. Liens hypertexte, aide en ligne : uniformisation et simplification des modes d'accès aux ressources pour en faciliter l'usage (qui n'est pas une donnée!!!).
6. Pas d'obligation de savoir tout résoudre (Voir [KOEGEL & Al. 89]). Le système doit surtout savoir ce qu'il ne sait pas, connaître ses limites.

Chapitre VI.4

Architecture de CAMELEON

Remarque : CAMELEON n'étant pas opérationnel, ce chapitre présente les idées générales de son implantation.

VI.4.a Simplifier, évaluer, ordonner, typer

L'un des problèmes les plus complexes au niveau du calcul algébrique est celui de la simplification. Il n'y a pas de forme canonique ([CAVINESS 70], [MOSES 71b]), i.e. :

- l'expression intéressante dépend du problème que l'on cherche à résoudre. Prenons par exemple, la fonction : $f(x) = \cos x * (x + 1)(x - 1)$. Cette expression est parfaite pour résoudre l'équation $f(x) = 0$, du fait qu'elle se présente comme un produit de facteurs. Quand on étudie la parité de f ou si on recherche une primitive de f, l'expression équivalente $\cos x * (x^2 - 1)$ est meilleure.
- on ne possède pas d'algorithme sur l'ensemble des expressions.

Cet écueil de simplification est central puisqu'il peut intervenir dans le cadre de tous les problèmes de calcul formel et peut s'avérer plus difficile à résoudre qu'un problème que l'on peut être amené à se poser sur une expression. CAMELIA maîtrise très mal la simplification (très souvent sous-traitée à REDUCE), ce qui rend son fonctionnement quelque peu incohérent. On peut remarquer que cette difficulté est considérablement diminuée quand on se limite au traitement des polynômes (bien qu'elle soit loin de disparaître).

Un problème voisin est celui de l'évaluation. Par exemple, quand on cherche à savoir si une expression ne contenant pas de fonctions transcendantes est un polynôme, il faut s'assurer que toutes les puissances des sous-expressions dominant une inconnue sont des entiers positifs.

Ainsi, $x^{(2^2-3)} + 1$ nécessite le calcul de $(2^2 - 3)$. Que faire de $(2^{34}) - 1$? Le fait que ce nombre est un entier positif est de l'ordre de l'évidence sans faire la moindre évaluation, ni même estimation du résultat. Le problème est plus difficile avec une machine.

Ces quelques remarques montrent qu'un résolveur capable d'expliciter sa démarche de recherche doit maîtriser relativement finement les problèmes de simplification et d'évaluation. Ces dernières ne peuvent pas être considérées comme de simples procédures externes, il faut

pouvoir les contrôler en fonction de l'objet courant et du but recherché (dans MATHPERT, on peut attacher une condition à un opérateur passé au 'simplifieur', de plus, une vingtaine de paramètres de contrôle affectent les opérations effectuées par ce simplifieur [BEESON 89]).

Si on représente les règles de simplification par des règles de réécriture, on peut ne chercher à simplifier que sous certaines contraintes. En particulier, un typage partiel des expressions peut filtrer les paquets de règles de réécriture à essayer (voir PRESS) :

- x, $minus(x)$, $\alpha * x$ sont de même type (monômes en x de degré 1) et on peut réduire des expressions sur des sommations.
- $x^2, \alpha * x, x^4, \ldots$ sont des monômes en x de degré différents, mais il y a des simplifications possibles sur des produits.

Le typage peut aussi s'avérer très utile pour filtrer les méthodes ayant des chances d'aboutir pour résoudre un problème.

De la même manière, ordonner les expressions est indispensable pour limiter l'espace de recherche (pour rapprocher, par exemple, les éléments de même type, opérateurs ou inconnues).

VI.4.b Les objets du système

Dans la suite, le terme 'objet' ne correspond pas complètement à ce que l'on entend dans les langages orientés objets.

Nous avons vu précédemment (voir VI.2, page 185) que l'une des limitations des résolveurs pédagogiques était liée à une connaissance insuffisante des objets mathématiques. Les idées qui sous-tendent CAMELEON consistent à déclarer des objets, leur associer des propriétés et baser la recherche d'une solution sur ces propriétés, i.e. relier des types de problèmes à des propriétés générales sur les objets mathématiques. La recherche ne se réduit pas à la vérification de certaines contraintes sur les objets courants pour déclencher des plans généraux, mais correspond à la déduction de l'ensemble des propriétés intéressantes dans le cadre du problème courant, devant permettre le choix raisonné d'une méthode appropriée. En fait, le système est basé non pas sur les opérateurs ou les méthodes, mais sur des concepts.

La représentation choisie tente de modéliser les connaissances à un niveau élémentaire (en gros le niveau Terminale). On distingue :

- des objets mathématiques (nombres, ensembles),
- des objets logiques (booléens, formules),
- des connecteurs (opérateurs),
- des problèmes,
- des actions.

On dispose aussi de variables, et d'expressions formées à partir de constantes (qui sont des objets d'un type précédent) et de connecteurs. (Consulter en annexe les propriétés associées aux divers objets).

Toute expression contenant une variable est une fonction numérique ou booléenne (suivant l'ensemble des valeurs prises par cette expression). (A priori, on ne considère pas d'expression construites à partir d'ensembles et contenant une variable). Une équation est une formule

contenant un seul opérateur de relation (=), les membres gauche et droit étant de type numérique (ou ensembliste).

Un objet est soit :

- une constante (2, e,...),
- une variable typée (X entier(positif), X booléen,...)
- une expression bien formée à partir des constantes, des variables et des connecteurs.

Remarque : les notions d'opérateurs et de fonctions se recouvrent. Ainsi, un opérateur comme Log peut aussi être vu comme la fonction Log, possédant certaines propriétés spécifiques (ensemble de définition, ensemble image, etc.). Le comportement d'un attribut ou de la valeur d'un attribut dans la composition avec l'opérateur Log peut s'étudier soit à partir d'une déclaration spécifique soit à partir des propriétés particulières de la fonction Log.

L'idée centrale de CAMELEON est d'étudier de manière exhaustive le comportement des propriétés des objets dans la construction des expressions. Ces propriétés sont reliées directement aux propriétés des connecteurs par des déclarations de stabilité. En fait, on associe à chaque opérateur son comportement vis-à-vis de tous les types d'objets et d'attributs.

Une propriété ou un type est dite stable pour un opérateur unaire, si l'application de l'opérateur à un objet ayant cette propriété conserve cette propriété. De même pour des opérateurs binaires ou variaires.

Exemples :
+ stable(entier, relatif, fraction réel)
+ stable(signe)
+ stable(polynôme)

On associe de la même manière des règles de réécriture sur les propriétés (ex. règle des signes générale). Des stabilités plus générales sont aussi reliées à certaines propriétés : paire (pour une fonction) stable pour tout opérateur.

Remarque : ce sont ces déclarations de stabilité qui permettent d'assurer un traitement ascendant des expressions, ces dernières pouvant être d'ailleurs entièrement déterminées ou définies avec des variables et des contraintes. On "remonte" ainsi des propriétés, ou on classe des sous-expressions maximales pour une propriété donnée. Il faut noter que l'enseignement des mathématiques tend à mélanger diverses interprétations et favorise peu les généralisations. Ainsi, dans le cadre de résolution des équations ou des inéquations, les justifications habituelles s'appuient sur des connaissances différentes, masquant souvent des invariants. Par exemple :
$(x + 2 = 3) \Rightarrow (x = 3 - 2)$ s'explique par le fait qu'on ne change pas l'égalité en ajoutant ou en retranchant un même nombre aux 2 membres.
$(e^x = 4) \Rightarrow (x = Log\ 4)$ paraît être une action totalement différente.

Dans les deux cas, c'est pourtant la même technique d'isolation (PRESS) qui est utilisée, i.e. application de la fonction réciproque. Cette explication rend mieux compte des difficultés associées aux fonctions non inversibles. De la même manière, pour les inéquations, les règles de changement de sens cachent l'aspect fondamental lié à la monotonie de la fonction réciproque.

VI.4.c Notion de problème

VI.4.c.1 Définition d'un problème

Un problème est un objet du système d'un type particulier. C'est un prédicat qui peut se représenter par un triplet (Attribut, Objet , Valeur) :

- l'attribut est déterminé, il s'agit d'une propriété d'un objet quelconque.
- l'objet est typé : il est connu soit par son expression, soit par des contraintes.
- la valeur est soit inconnue, soit connue, soit une formule.

Par exemple, si on connaît les objets F, fonction impaire, et N entier : (parité, F^{2N}, paire) est un problème.

- parité est un attribut des fonctions réelles,
- F^{2N} est une fonction réelle définie par une expression,
- paire est la valeur de l'attribut parité.

(parité, F^{2N},?) est un problème, la valeur de l'attribut considéré est ici inconnue.

(parité, F^{2N}, non impaire) est un problème, on doit ici uniquement vérifier une contrainte sur la valeur. Dans la résolution, si on suppose F continue, il peut suffire de calculer l'image de 0 pour conclure.

(limite (0), x^N, entier positif) est un problème, où l'attribut comporte un argument (limite au point 0) et la valeur est un type d'objet.

Il n'est pas possible ici de chercher un objet connaissant la valeur d'un ou plusieurs attributs. Ainsi, on ne peut pas résoudre des questions telles que : trouver une fonction impaire et croissante. On peut cependant vérifier si les contraintes sont compatibles : trouver une fonction paire et croissante devrait amener à la solution fonction constante.

La définition précédente d'un problème englobe aussi les formes : (Action, Objet, Contraintes)

- Action : simplifier, ordonner, évaluer (ceci correspond à trouver une nouvelle expression de l'objet considéré).
- Contraintes : But courant, filtrage des méthodes, etc.

En particulier le triplet (résoudre, problème, contraintes) est lui-même est problème.

VI.4.c.2 Attributs d'un problème

Un problème possède aussi certaines propriétés ou attributs. Une première idée est de considérer la notion de complexité. Ceci est indispensable dans les résolveurs de type système expert pour choisir les règles ou les plans de résolution qui semblent le plus adaptés au problème courant. Ainsi CAMELIA utilise des variables COUT/ESPOIR attachées à chaque plan, et le classement des plans possibles s'effectue à l'aide de ces valeurs et d'une mesure de la complexité de l'expression courante. Cette mesure ne peut d'ailleurs être calculée que si l'expression est entièrement connue.

La question que l'on peut se poser est de savoir si l'on peut connaître A PRIORI la complexité d'un problème donné, en fonction de la complexité de l'expression que l'on considère.

Exemples :

1. $(e/\pi \in Q)$ ou $(e/2e \in Q)$
   Les deux expressions sont très courtes, et semblent à première vue équivalentes au niveau complexité. Le premier problème est cependant quelque peu résistant, tandis que le deuxième est évident moyennant une petite simplification.
2. $f(x) = (x+1)^3 - (x-1)^3$
   $g(x) = Log\ (|x| * e^{|x|} * x^2 * sin^2 x * cos x)$
   On admettra volontiers que la fonction g est plus complexe que la fonction f. Si on cherche à démontrer la parité de ces deux fonctions, le cas de g est cependant immédiat alors que celui de f nécessite quelques calculs.

En fait, ces deux exemples montrent qu'il n'est en général pas possible de déterminer la complexité d'un problème avant de le résoudre effectivement. Seule une définition duale a posteriori peut convenir, i.e. mesurer la complexité de la solution (ou d'une des solutions) quand on peut la trouver. D'ailleurs, la complexité d'un problème est tout à fait relative aux méthodes de résolution que l'on peut utiliser. Sans aucune connaissance, on ne peut résoudre aucun problème. Cependant, la même chose s'applique même si l'on suppose que la personne censée résoudre possède une très large culture mathématique.

Remarque : savoir si un problème est difficile nécessite une très bonne connaissance des mathématiques. Certaines conjectures que l'on peut proposer peuvent se résoudre très simplement, d'autres peuvent se révéler extrêmement résistante. Un mathématicien confronté à un problème, essaie d'abord de le résoudre lui-même. S'il n'y parvient pas, il demande à l'un de ces pairs, ensuite, il consulte un spécialiste reconnu. Si toutes ces étapes sont franchies, on considère que la conjecture est acceptable et on peut la proposer à la sagacité de la communauté mathématique. Cette manière de procéder montre bien que la difficulté attachée au problème est liée à une incapacité à le résoudre rapidement. Dans le cadre scolaire, les exercices d'application sont des faux problèmes, puisqu'ils contiennent leur solution. On retrouve les difficultés des problèmes nécessitant la sélection de méthodes (voir VI.3.b, page 194).

Cette définition de la complexité d'un problème nécessite de définir une relation d'ordre sur les solutions. Une solution correspond à une suite finie de transformations de l'expression et d'applications de méthodes. La taille des calculs rend mal compte de la difficulté d'un problème, de même la comparaison des méthodes entre elles ne fournit pas de hiérarchie évidente. Dans un calcul de primitives, un changement de variable est-il plus compliqué qu'une intégration par parties? En conséquence, définir un ordre total entre les solutions est aussi exclu. Par contre, des ordres partiels sont envisageables.

Si on a deux problèmes P1 et P2 auxquels sont associées les solutions S1 et S2, et si S1 est inclus dans S2, on peut dire que P2 est plus complexe que P1 (modulo peut-être les méthodes utilisées, une autre solution de P1 ou P2 pouvant être d'une nature différente).

Remarque : Il y a une forte analogie entre la notion de complexité attachée à un problème et la notion de gravité liée à une erreur (voir V.3.a.1, page 171). Cette dernière est une trace d'un cheminement qu'on essaye de reconstituer. Ce n'est qu'un symptôme non une maladie. C'est l'analyse que l'on porte sur les causes de l'erreur et sur celui qui l'a produite qui conduit à un diagnostic de gravité. L'erreur n'est pas en elle-même grave (dans le cadre où on se place, il n'y a pas d'effet de bord, ce n'est bien sûr pas le cas quand on apprend à conduire dans les conditions réelles!), elle ne l'est que par une

interprétation. En ce qui concerne un problème, c'est le cheminement à faire, et celui qui le fait qui conditionne la difficulté. Dans les deux cas, cela réfère aux connaissances maîtrisées par la personne qui résout.

Un premier attribut associé à un problème est la liste de ses solutions. D'autres attributs sont utiles :

- cohérence (de type booléen) :
- résolvabilité (de type booléen)
- valeur de vérité (de type booléen)

Un problème peut être faux (exemple (parité, x, paire)). Il peut être vrai, sans que l'on connaisse une solution : problème connu (axiome). Comme le note R.Cuppens, un problème peut être vrai ou faux suivant la théorie admise (ex. : axiome des parallèles).

VI.4.c.3 Relations et opérateurs sur les problèmes

A partir du filtrage sur les expressions, on peut définir une relation d'ordre partiel :
une expression $E1$ est plus générale qu'une expression $E2$ s'il existe une substitution σ telle que $\sigma(E1) = E2$.
Ceci correspond sur les expressions à choisir des valeurs particulières à certaines variables, ou à ajouter de nouvelles contraintes.

Exemples: x^N avec N entier positif est plus général que x^3
x^P avec P réel est plus général que x^N avec N entier
(dans ce deuxième exemple, l'expression est la même, mais les contraintes sont moins fortes sur le type plus général)

De la même manière, un problème P1 sera plus général qu'un problème P2 si P1 = (attribut, O1, valeur) et P2 = (attribut, O2, valeur) où les attributs et les valeurs sont les mêmes et O1 est plus général que O2. (O1 et O2 étant des objets de même type).

En s'inspirant de cette relation, on voit comment particulariser ou généraliser un problème, en particularisant ou généralisant l'expression de l'objet considéré.

D'ailleurs si S1 est une solution d'un problème P1 et que P1 est plus général que P2, $\sigma(S1)$ est automatiquement solution de P2.

Un deuxième type de relation entre les problèmes est lié aux valeurs des attributs. Si on a deux valeurs (formules) V1 et V2, V1 est plus générale que V2 si tout objet vérifiant V2 vérifie

	V1 = entier positif	V2 = 2
aussi V1 :	V1 = réel	V2 = rationnel
	V1 = paire ou impaire	V2 = paire

Alors, si P1 = (attribut, O, V1) et P2 = (attribut, O, V2) avec V1 plus générale que V2, on dira que P2 est plus spécifique que P1.

Généraliser un problème peut conduire à généraliser l'objet, et à généraliser la valeur pour garder la vérité :
P1 = (parité, x, impaire)
P2 = (parité, x^N, paire ou impaire)

Remarque : On mélange ici les notions d'instanciation et de typage (correspondant à l'appartenance et l'inclusion), ce qui pourrait s'avérer gênant dans certaines occasions. Cependant, cette confusion est pratique, dans un souci d'uniformisation, pour traiter de manière identique des objets connus par une expression et des objets connus uniquement par un type et des contraintes.

Un troisième type de relation concerne les liens entre des problèmes portant sur des attributs différents. C'est une forme d'implication : implique ((parité, f, paire ou impaire), (def, f, symétrique))
implique ((parité, f, impaire) et (continue(0), f, vraie), (valeur(0),f,0))
implique ((type, f, polynôme), (continuite,f,R))
implique ((continuite,f,R), continue(Y),f,vraie))

Dans la résolution d'un problème, ces implications peuvent être utiles pour montrer que le problème est faux.

Les problèmes sont des prédicats, et on peut utiliser les connecteurs logiques habituels : et, ou, non, (⇒ vient d'être vu). On construit ainsi des expressions formée de conjonctions, disjonctions et négations de problèmes élémentaires.

Remarque : l'aspect 'représentation' est souvent primordial pour suggérer de bonnes méthodes de résolution. On devrait pouvoir associer à un même problème, des contextes de résolution très différents.

Par exemple :
Soient deux réels positifs x et y tels que $x < y$, montrer que :
$x < \frac{x^2+y^2}{x+y} < y$
Ce problème se résout à l'aide de calculs et de majorations dans R :
$x = \frac{x(x+y)}{x+y} = \frac{x^2+xy}{x+y} < \frac{x^2+y^2}{x+y}$ car $0 < x < y$ On peut aussi le rattacher à l'idée de barycentre des deux points : le barycentre des points M(x) et N(y) affectés des coefficients positifs x et y est le point $G(x^2 + y^2/x + y)$ qui est situé entre les points M et N.

VI.4.c.4 Notion de sous-problème

Au cours de la résolution d'un problème, on est amené à poser de nouveaux problèmes qui interviennent comme des sous-problèmes du problème initial suivant une méthode de résolution. Ces sous-problèmes sont reliés plus ou moins directement au problème initial, et leur résolution peut être plus ou moins ardue. En particulier, il faut éviter de tenter de résoudre un sous-problème plus difficile que le problème lui-même. Comme il n'y a pas de mesure a priori de la complexité d'un problème, cela nécessite une forme de contrôle des recherches successives suivant les buts poursuivis. Les résolveurs travaillant en profondeur d'abord évitent difficilement cet écueil. C'est l'exemple de CAMELIA, d'autant plus que la richesse d'expression des plans, autorisant les conditions-problèmes, peut conduire à des recherches complexes et inutiles (montrer qu'une fonction est paire avant d'effectuer un changement de variable pour un calcul de primitives).

Dans certains cas, on arrive à un sous-problème équivalent au problème initial. Par exemple, en cherchant à résoudre le problème P1 :
P1 = (parité, cos x * (x + 1) (x - 1), paire)
remarquant que la fonction x -¿ cos x est paire, on est conduit au problème P2 :

P2 = (parité, (x + 1) (x - 1), paire)
P2 est le même problème que P1, mais sur une sous-expression, et on sait qu'il y a une équivalence, i.e. P1 est vrai si et seulement si P2 est vrai. Cette équivalence permet de faire des réductions sûres dans l'arbre de recherche d'une solution.

Remarque : dans l'exemple précédent, l'équivalence entre P1 et P2 est un peu abusive, la fonction cos ayant des zéros. Un argument de continuité (sur les fonctions) doit cependant l'assurer!

VI.4.d Fonctionnement général du système

VI.4.d.1 Les actions

Les actions sont des objets spécifiques du système. On dispose d'actions élémentaires et d'expressions formées à partir de celles-ci et de connecteurs spécifiques (et, ou, si, alors, sinon, séquence,...). De telles expressions correspondent à des plans. Une action est liée à un couple (Attribut, Objet). Le déclenchement d'une action dépend des valeurs des deux éléments de ce couple : toute action pose problème!

On distingue des actions de type :

- système : sauvegarder, charger, gestion écran (afficher), etc.),
- méta-mathématique : créer objet (SOIT), noter, supprimer,...,
- mathématique : changer de variable, simplifier, ordonner, factoriser, évaluer, résoudre...

Les règles de réécriture ($Log\ a*b = Log\ a+Log\ b$) sont attachées aux connecteurs, les méthodes générales et plans sont liés aux problèmes. L'aboutissement de chacune des actions permet de préciser la valeur d'un argument d'un objet (autres expressions, divers typages, etc.).

On a une dualité problème / action. Au niveau méta, on associe un plan général au problème de type 'problème'. Ceci permet de créer des types de méta-règles instanciables associées à l'objet problème : Sur un objet d'ordre supérieur à 0 (i.e. contenant une ou plusieurs inconnues), essayer des valeurs particulières pour trouver un contre-exemple ou préciser la valeur cherchée (ex. dans un problème de recherche de parité). En fait, cette instanciation réfère à l'attribut du problème courant.

On peut ainsi décrire de manière générale divers modes de résolution :

- méthode directe,
- contraposée/contradiction,
- récurrence,
- contre-exemple.

Remarque : d'une manière plus précise, les actions sont liées à un but, ce qui introduit des contraintes sur la valeur cherchée. Ainsi, ordonner et simplifier correspondent à l'attribut expression d'un objet, la nouvelle expression cherchée devant satisfaire à certaines conditions. (A préciser sur ces actions ou méta-actions qui sont liées à la forme expression générique plus qu'aux objets).

Une quantification minimale des actions est nécessaire pour savoir si on peut les déclencher. Le premier tri peut consister simplement à séparer les actions immédiates des actions non immédiates. Les premières sont lancées automatiquement, les résultats obtenus n'étant pas forcément les meilleurs possibles. Ainsi typer une expression peut conduire de manière immédiate à un résultat flou du style fraction rationnelle ou polynôme (un calcul intermédiaire n'étant pas évident). Dans un premier temps, on peut considérer comme évident tout ce qui se déduit automatiquement des propriétés de stabilité des opérateurs.

VI.4.d.2 Les interpréteurs

Deux interpréteurss généraux sont construits :

- système générique de traitement d'expression,
- un gestionnaire de plans.

Ce dernier gestionnaire différencie des démarches de réduction sûres et non sûres.

VI.4.d.3 La résolution

Il faut remarquer que les actions principales (simplifier, ordonner, évaluer, typer) sont généralement imbriquées les unes dans les autres, leur déclenchement ne pouvant en aucun cas être irrévocable. Le raisonnement général n'est ni en profondeur ni en largeur, il se développe plutôt par strates successives : faire tout ce qui est immédiat, analyser les résultats obtenus, si le problème n'est pas terminé, choisir et appliquer toutes les méthodes possibles jusqu'à trouver une solution. Dans l'application d'une méthode, on s'arrête temporairement si on tombe sur un problème annexe non résoluble immédiatement. En fait, l'idée fondamentale consiste à introduire une phase entre :

- l'expression initiale du problème,
- le choix d'une méthode de résolution.

Cette phase correspond à une recherche systématique de toutes les propriétés a priori utiles pour la résolution du problème courant.

Voici le plan de résolution attaché à l'objet 'problème' :

1. Rechercher une 'évidence' :
   - résultat connu,
   - cas d'impossibilité,
   - "évidence perceptive", (tous les termes contenant X pairs pour une étude de parité)
2. Collecter toutes les informations disponibles et voir leur influence. (en particulier, prendre la représentation de l'expression la plus intéressante).
3. Choisir une méthode de réduction sûre (se ramener à des sous-problèmes équivalents) (éviter le retour arrière)
4. Vérification éventuelle de cohérence (vérifier sur des cas particuliers).
5. Lister les méthodes applicables et les trier
6. Lancer éventuellement ces méthodes.

L'annexe 6 détaille un plan de résolution pour la recherche de la parité d'une fonction (voir annexe F, page 259).

Septième Partie

Conclusion

Le parti pris ici de faire une rédaction en hypertexte amène à considérer différents points de vue et encourage le développement de controverses qui peuvent être productives. En ce qui concerne les problématiques développées dans le cadre EIAO, cette thèse permet de dresser certains bilans :

- les architectures proposées pour la conception des ITS sont inadaptées pour la réalisation d'outils intégrables dans le cadre scolaire formel, au moins dans son organisation actuelle;
- les études centrées sur la modélisation de l'apprenant sont plus performantes pour les systèmes d'aides intelligents intégrés à des tâches déjà informatisées;
- les 'diagnostiqueurs', reconstruisant un comportement de manière fine à partir d'une déviation des connaissances expertes, semblent destinés plutôt à être un outil de réflexion pour les enseignants. Les modèles de génération des erreurs n'ont pas encore la maturité suffisante, de même que le diagnostic cognitif basé sur les conceptions profondes, pour générer des produits véritablement opérationnels;
- la gestion du curriculum, l'accès contrôlé à des bases d'informations correspond plus à des besoins affirmés dans la documentation.
- la prise en compte de l'environnement, du cadre d'utilisation (externe à la machine), et le fait que les techniques d'intelligence artificielle ne permettent de maîtriser que des univers extrêmement réduits, militent en faveur d'environnements d'apprentissage associés à des tuteurs locaux destinés plus à l'entraînement, au renforcement, à la remédiation ponctuelle, à la performance (formation professionnelle) qu'à l'acquisition de notions.

En conclusion, ce sont les environnements d'apprentissage qui semblent le mieux se prêter à un cadre scolaire. Dans l'enseignement primaire et au début du collège (passage concret-formel), il semble souhaitable de développer des micromondes favorisant ce passage (le paradigme des deux mondes est prometteur) ainsi que des activités de programmation.

L'aspect prédominant du résolveur est présent dans toutes les parties, et on peut rappeler la diatribe forme / fond, qui revêt de multiples apparences :

interface	/	connaissances
exposition	/	raisonnement
point de vue	/	compréhension (indices)
application	/	sélection
syntaxe	/	sémantique
algorithme	/	sens de l'opération

Trois types de projets peuvent être dégagés :

- création d'un environnement intégré pour l'apprentissage de la géométrie : aide à la réalisation de la figure, aide à la recherche et aide à la rédaction. Il faut noter qu'un tel projet n'a de sens que si on peut assurer une représentation interne partageable entre tous ces environnements.
- structuration des connaissances mathématiques par :
  - la réalisation de résolveurs (les plus déclaratifs possibles),
  - la gestion du curriculum en hypertexte.
- analyse de l'impact des outils de diagnostic.

Enfin, comme il n'y a pas encore de système prenant en compte toutes les contraintes d'un tuteur intelligent, on peut remarquer que, dans ce travail, l'exploration des différentes composantes s'est effectuée au travers de plusieurs systèmes. Elle est résumée dans le tableau suivant :

Hypertexte	ARRIA	BADAUD	CAMELEON
Environnement	Enseignant	Modèle élève	Domaine
Non résolveur	Semi-résolveur	Résolveur perturbable	Quasi-résolveur
Accès			
Analyse domaine + utilisateur	Scénario pédagogique	Protocoles	Expert
AVAL	AVAL	COURANT	AMONT

Annexe A

Structure de l'hypertexte

Le texte de cette thèse est aussi accessible dans un format hypertexte, cette annexe précise certains points techniques de cette réalisation.

Tout d'abord, on peut remarquer que l'hypertexte est partiellement redondant avec le document imprimé, mais il est complémentaire et offre quelques avantages au niveau de la consultation. Ensuite, il faut noter que, si le travail imprimé est une forme figée qui n'est plus modifiée, l'hypertexte peut être constamment complété et enrichi.

Le choix des outils pour la constitution de l'hypertexte répond à plusieurs contraintes :

- possibilité d'accès à un maximum de personnes, ce qui suppose son implantation sur des machines très répandues, avec, au moins, un système de lecture de l'hypertexte peu onéreux,
- facilité de lecture et d'impression,
- possibilité de lancement de programmes,
- possibilité de récupération de textes au format ASCII,
- facilité de codage des liens,
- possibilité de récursion dans les notes.

Pour répondre à ces diverses demandes, le choix de la machine cible s'est porté sur un compatible PC, avec les programmes **Connexion** (Hatier) et **HyperInfo** (Softia). Une version HyperCard pourra aussi peut-être être développée, mais elle nécessitera le transfert de tous les textes et un nouveau codage des liens (elle ne permettra cependant pas le lancement des programmes qui ont été conçus pour des compatibles PC).

Le mélange des deux programmes Connexion et HyperInfo permet d'avoir une structure riche mélangeant plusieurs types de liens :

- renvoi entre les parties du texte,
- renvoi à une problématique générale, ce qui donne une forme étoilée (voir III.1.a.3.d, page 84) avec des liens bi-directionnels entre des thèmes généraux et diverses parties du document. Par exemple, la dichotomie forme / fond, ou le concept de micromonde.
- renvoi à la bibliographie :

  - nom d'auteur ou référence,
  - nom de produit : un récapitulatif général des systèmes mentionnés sera ainsi directement accessible,
- lancement (programme, images, etc...),
- renvoi à un commentaire (collabaration d'une autre personne).
- table des matières et consultation d'index.

Ce travail fait l'objet d'une édition séparée.

Annexe B

Les micromondes Logo - l'exemple de PROD

B.a Généralités

Il semble inutile de décrire en détail les divers micro-mondes réalisés et les techniques utilisées. On ne s'intéressera ici qu'à l'un d'entre eux, PROD, pour plusieurs raisons :

- implantation d'un environnement déclaratif en LOGO comprenant :
  - un langage d'expression avec éditeur spécialisé,
  - un interprète,
  - des extensions vers l'EAO traditionnel.
- exemple de micromonde sur la grammaire,
- système-expert scolaire dépassant le cadre de la classification.

En ce qui concerne le dernier point, on peut remarquer, à ma connaissance, que PROD est le seul exemple d'environnement permettant à des enfants de travailler en déclarant des règles de production, dans un cadre non trivial. PROD n'est cependant que partiellement déclaratif, puisque l'ordre des règles intervient de façon déterminante. Ceci est d'ailleurs plutôt une simplification puisqu'il n'y a en général qu'une seule solution et que cela évite le recours à des conditions exclusives.

B.b Le langage

On distingue deux parties : ce qui est lié à la gestion générale et au système de production (le moteur, l'édition, l'affichage, la liaison avec le disque) et les primitives permettant de travailler dans le domaine spécifique de la grammaire, i.e. le langage utilisé dans l'écriture des règles.

B.b.1 Primitives de gestion :

`def` (nom) : permet de déclarer une nouvelle fonction. Ainsi `def "fem` définit la fonction LOGO fem de code suivant :

```
POUR fem :L
RENDS PHMAP [RECHERCHE "fem] :L
FIN
```

Les listes (nom).EX et (nom).RG sont aussi créées automatiquement. La première est une liste d'association qui correspond à l'ensemble des cas particuliers, la seconde correspond à l'ensemble des règles de production.

On trouve ensuite des fonctions de service très classiques :

`def?` (nom) : prédicat qui renseigne sur l'existence de (nom).

`efface` (nom) : efface la procédure spécifiée ainsi que les deux listes associées.

`garde` (nom) (liste) : sauvegarde dans un fichier nommé (nom).pro toutes les déclarations associées à chacune des fonctions de la liste.

`ram` (nom) : ramène en mémoire le contenu du fichier (nom).pro

`ed` (nom) : permet de déclarer ou de modifier une liste de cas particuliers ou de règles.

`ed1` : permet de reprendre les modifications ou déclarations en cas de sortie intempestive de l'éditeur.

`affiche` (nom) : affiche sur l'écran l'ensemble déclarations concernant (nom).

`afr` (nombre) (nom): affiche la règle numéro (nombre) de (nom).

`trace` : mode qui permet la visualisation de la recherche

`detrace` : retour au mode normal

`explique` (liste) : où (liste) correspond à un appel à une fonction de recherche, permet de fournir la solution ainsi que la manière dont elle a été obtenue.

`?` : affiche la liste des primitives de l'utilisateur.

B.b.2 Ecriture des règles

B.b.2.a générale

L'environnement est conçu pour faciliter l'écriture des diverses déclarations et se rapprocher du mode d'expression habituel. Il s'agit pour l'enfant de traiter un problème de grammaire, non de jongler avec une syntaxe ardue. La déclaration des cas particuliers s'effectue en donnant, ligne à ligne, le nom choisi et son transformé (ex. `doux douce`). Les règles sont formées d'une partie condition et d'une partie action.

Toutes les primitives travaillent sur une variable implicite qui est leur dernier argument (il est inutile de le préciser dans l'écriture des règles ce qui évite le recours aux variables). Les primitives fournies, ajoutées aux primitives classiques du Logo sont redondantes ce qui permet de choisir entre plusieurs écritures suivant son propre goût (en particulier, `remplace` utilisé éventuellement avec un joker peut se substituer à toutes les autres fonctions de construction).

B.b.2.b Prédicats

Ils s'utilisent dans la partie condition des règles. Une condition commence impérativement par le mot «si» ou contient uniquement le mot «sinon» (pour la dernière règle qui est alors automatiquement vérifiée). La partie condition peut être formée d'une conjonction de conditions, chacune d'entre elles étant séparée par le mot «et» (le «si» est facultatif après le «et»).

`term? (mot ou liste)` : rend VRAI si le mot se termine par la ou l'une des terminaisons choisies (cela permet de traiter la conjonction «ou»).

`elem? (liste)` : correspond à MEMBRE? après permutation des arguments

`dif? (mot)` : correspond à NON EGAL?

Remarque : Frédéric Robert a repris l'environnement pour ajouter la possibilité d'introduire un «sauf si» pour écrire des règles du type (pour le pluriel) :

```
si term? "ou
alors ajoute "s
sauf si elem? [chou bijou caillou genou pou joujou]
alors ajoute "x
```

B.b.2.c Fonctions de construction de mots

Ces fonctions sont utilisées dans la partie action des règles :

`ajoute (mot)` : correspond à MOT en permutant les arguments.

`remplace (mot1) (mot2)` : substitue la terminaison (mot2) à la terminaison (mot1) dans le mot courant.

`ote (mot ou nombre)` : retire la terminaison (mot) ou le (nombre) de lettresspécifiées.

`subst (nombre) (mot)` : substitue la terminaison (mot) au (nombre) de lettres spécifiées.

`dd` : ajoute au mot sa dernière lettre (équivalent avec le joker * à remplace "* "**).

B.b.2.d Les jokers

Les jokers sont des caractères spéciaux que l'on peut utiliser avec le prédicat `term?` et les fonctions de construction de mots. Ces jokers remplacent une lettre quelconque et restent liés à cette lettre pour une règle donnée. Par exemple, pour le féminin :

```
si term? [on ien el ul eil et s]
alors remplace "* "**e
```

B.b.2.e fonctions

`voyelle` : retourne la liste des voyelles

`consonne` : retourne la liste des consonnes

`rien` : dans la partie action des règles, indique que le mot ne subit aucune modification.

B.c L'interprète général

L'application de fem (voir B.b.1, page 214) consiste à appliquer la fonction RECHERCHE au mot :L ou à tous les items de la liste :L

Les déclarations, cas particuliers et règles, sont associées respectivement aux étiquettes fem.EX et fem.RG. La fonction RECHERCHE traite tout d'abord la liste fem.EX pour vérifier si le mot courant est un cas particulier. La réussite interrompt la recherche courante, sinon, la procédure PRODUC est lancée :

```
POUR RECHERCHE :NR :#MOT
DONNE "#M ASSOC :#MOT CHOSE MOT "EX. :NR
SI :TRACE [EC PH :NR [: cas particuliers.]]
SI NON VIDE? :#M [NOTE PH :NR 0 RENDS DER :#M]
RENDS PRODUC CHOSE MOT "RG. :NR 1
FIN
```

La procédure PRODUC exécute itérativement chacune des parties conditions des règles jusqu'à ce que l'une d'elles soit vérifiée.

```
POUR PRODUC :R :N
SI VIDE? :R [E PH [Aucune règle applicable pour] :#MOT RENDS "????]
EFT :JOKER
SI :TRACE [EC [Non applicable.] EC [] EC PH PH :NR [: règle numéro] :N]
SI EX TRAITE PREM PREM :R [] [NOTE PH :NR :N RENDS #EX DER PREM :R]
RENDS PRODUC SP :R SOMME 1 :N
FIN
```

TRAITE transforme la condition courante en une liste de prédicats exécutables

```
POUR TRAITE :L :LST
SI VIDE? :L [RENDS MP :LST []]
SI EGAL? PREM :L "et [RENDS PH MP :LST [] TRAITE OT "si SP :L []]
RENDS TRAITE SP :L MD PREM :L :LST
FIN
```

EX exécute chacune des conditions une à une :

```
POUR EX :COND
SI VIDE? :COND [RENDS "VRAI]
SI EGAL? :COND [[]] [RENDS "VRAI]
SI #EX PREM :COND [RENDS EX SP :COND]
RENDS "FAUX
FIN
```

#EX permet l'exécution d'un prédicat ou d'une action élémentaire.

```
POUR #EX :L
RENDS EXEC MD RECOTE :#MOT :L
FIN
```

La procédure explique initialise la variable #EXP et met NOTE à VRAI. A l'issue de la recherche, le contenu de #EXP est affiché :

```
POUR explique :L
LOCALE "NOTE LOCALE "#EXP
DONNE "NOTE "VRAI DONNE "#EXP []
EC PH [C'est] EXEC :L
#MONTRE :#EXP "E
FIN
```

La procédure NOTE garde la règle utilisée pour conclure :

```
POUR NOTE :L
SI NON :NOTE [STOP]
SI EGAL? 0 DER :L [DONNE "#EXP MD PH PREM :L [: cas particulier] :#EXP STOP]
DONNE "#EXP MD PH PREM :L PH [: règle numéro] DER :L :#EXP
FIN
```

B.d Exemples :

(Les exemples sont loin de résoudre complétement le problème!)

```
?affiche "fem
Cas particuliers : [gentil gentille] [bas basse] [épais épaisse]
 [frais fraîche] [doux douce] [faux fausse] [roux rousse] [beau belle]
 [nouveau nouvelle] [fou folle] [mou molle] [vieux vieille]

Règle numéro 1
si term? "e
alors rien

Règle numéro 2
si elem? [complet inquiet désuet discret]
alors subst 2 "ète

Règle numéro 3
si elem? [palot vieillot sot]
alors ajoute "te

Règle numéro 4
si term? [on ien el ul eil et s]
alors remplace "# "##e

Règle numéro 5
si term? "x
alors remplace "x "se

Règle numéro 6
si term? "f
alors remplace "f "ve

Règle numéro 7
si term? "c
alors ajoute "he
```

```
Règle numéro 8
sinon
ajoute "e

?affiche "adv
Cas particuliers : [présent présentement] [assidu assidûment]
  [gentil gentiment] [goulu goulûment]

Règle numéro 1
si term? voyelle
alors ajoute "ment

Règle numéro 2
si term? [ant ent]
alors remplace "*nt "*mment

Règle numéro 3
sinon
ajoute "ment fem
```

Exemples d'exécution :

```
?explique [adv "discret]
C'est discrètement
adv : règle numéro 3
fem : règle numéro 2

?afr 4 "adv
sinon
ajoute "ment fem

?afr 2 "fem
si elem? [complet inquiet désuet discret]
alors subst 2 "ète

?explique [adv "beau]
C'est beaument
adv : règle numéro 1

?trace
?ec fem "gentil
fem : cas particuliers.
gentille
?ec fem "heureux
fem : cas particuliers.
Non applicable.

fem : règle numéro 1
Non applicable.

fem : règle numéro 2
Non applicable.
```

```
fem : règle numéro 3
Non applicable.

fem : règle numéro 4
Non applicable.

fem : règle numéro 5
heureuse
?detrace
?explique [adv "vif]
C'est vivement
adv : règle numéro 3
fem : règle numéro 6
```

B.e Un exemple de questions/réponses

Bien que PROD soit typiquement un environnement destiné à la déclaration de règles, on peut facilement l'intégrer dans un logiciel de type questions réponses classique. Néanmoins, on peut en conserver les spécificités :

- choix d'une liste de mots,
- explication d'une solution obtenue,
- travail sur les règles déclarées.

En particulier, l'ordinateur ne détient pas la vérité mais se contente d'appliquer les règles qu'on lui a fournies. Ces dernières peuvent être fausses!

On part d'une liste de mots déclarés séparemment et stockés dans la liste nommée `#LISTE`.

```
POUR test :#ED :#LISTE
SI NON def? :#ED [STOP]
QUEST :#LISTE 0 0
FIN

POUR QUEST :LST :T :B
SI VIDE? :LST [TOTAL STOP]
LOCALE "NOTE LOCALE "#EXP LOCALE "CHOIX
EC [] DONNE "CHOIX CHOIX :LST
EC PH PH PH :#ED [de] :CHOIX [?]
DONNE "NOTE "VRAI DONNE "#EXP []
QUEST OTER :CHOIX :LST :T + 1 SOMME COMP? LL EXEC PH :#ED MOT "" :CHOIX :B
FIN

POUR COMP? :L1 :L2
SI NON VIDE? :L1 [SI EGAL? PREM :L1 :L2 [EC [J'ai la même réponse.] RENDS 1]]
EC PH PH [La réponse que l'on m'a donnée est:] [-->] :L2
#REGLE :#EXP RENDS 0
FIN

POUR CHOIX :L
```

```
RENDS ITEM 1 + HASARD COMPTE :L :L
FIN

POUR TOTAL
EC [] EC []
EC PH PH [Pour] :T [questions :]
TAPE PH PH [nous sommes d'accord sur] :B [réponse]
SI :B > 1 [EC [s.]] [EC []]
FIN

POUR OTER :L :LST
SI EGAL? :L PREM :LST [RENDS SP :LST]
RENDS MP PREM :LST OTER :L SP :LST
FIN

POUR #REGLE :L
SI VIDE? :L [EC [] STOP]
EC PREM :L
SI NOMBRE? DER PREM :L [afr DER PREM :L PREM  PREM :L]
#REGLE SP :L
FIN
```

Exemple d'exécution :

```
?test "fem :TEST
```

fem de sot?
sote
La réponse que l'on m'a donnée est: sotte
fem : règle numéro 3
si elem? [palot vieillot sot]
alors remplace "# "##e

fem de gentil?
gentile
La réponse que l'on m'a donnée est: gentille
fem : cas particulier

fem de discret?
discrete
La réponse que l'on m'a donnée est: discrète
fem : règle numéro 2
si elem? [complet inquiet désuet discret]
alors subst 2 "ète

fem de orange?
bleue
La réponse que l'on m'a donnée est: orange
fem : règle numéro 1
si term? "e
alors rien

Annexe B. Les micromondes Logo - l'exemple de PROD

fem de franc?
france
La réponse que l'on m'a donnée est: franche
fem : règle numéro 7
si term? "c
alors ajoute "he

fem de vif?
vife
La réponse que l'on m'a donnée est: vive
fem : règle numéro 6
si term? "f
alors remplace "f "ve

Pour 6 questions :
nous sommes d'accord sur 0 réponse

?test "adv :TEST
adv de gentil?
gentillement
La réponse que l'on m'a donnée est: gentiment
adv : cas particulier

adv de franc?
francement
La réponse que l'on m'a donnée est: franchement
adv : règle numéro 3
sinon
ajoute "ment fem

fem : règle numéro 7
si term? "c
alors ajoute "he

Annexe B. Les micromondes Logo - l'exemple de PROD

Annexe C

Notion de point de vue

C.a APILOG

C.a.1 Points de vue et explications

Plusieurs lectures d'un programme PROLOG :

L'exemple suivant est une partie du code Prolog d'un jeu de mastermind. Il illustre l'utilisation de la notion de point de vue. Le même bout de programme sert à chercher un code, compatible avec un essai précédent et le score associé, un code, connaissant un essai et le score associé ou le score en comparant l'essai proposé et le bon code.

En fait, dans le prédicat mm, un des trois paramètres est inconnu tandis que les deux autres sont instanciés. Toutes les requêtes peuvent être lancées par le prédicat mm. Voici quelques exemples de requêtes :

```
       Choisissez une requête

1 - mm([rouge,vert,bleu,jaune],Code,[[i,i],[i]])?
2 - mm(Essai,[rouge,vert,bleu,jaune],[[i,i],[i]])?
3 - mm([rouge,vert,bleu,jaune],[jaune,vert,vert,bleu],Score)?
4 - mm([rouge,vert,bleu,jaune],Code,Score)?
5 - mm(Essai,[rouge,vert,bleu,jaune],Score)?
```

Recherche du **code** d'un essai du score

```
codepossible([],_).
codepossible( [[P1,S1]|Ps],Co):-
    mm(P1,Co,S1),codepossible(Ps,Co).

score(Essai,Total):-
  code(Bcode),
  mm(Essai,Bcode,Total).

mm(Essai,Code,[Bp,Mp]):- bplace(Essai,Code,Ressai,Rcode,Bp),
                         mplace(Ressai,Rcode,Mp).
```

```
bplace([],[],[],[],[]).
bplace([U|P],[U—C],P1,C1,[i|S]):-
    bplace(P,C,P1,C1,S).
bplace([V|P],[W—C],[V|P1],[W|C1],S):-
    diff(V,W),bplace(P,C,P1,C1,S).
```

```
mplace([],C,[]):-!.
mplace([U|P],C,[i|S]):-
  del(U,C,C1),!,mplace(P,C1,S).
mplace([U|P],C,S):-
  not(member(U,C)),mplace(P,C,S).
```

Recherche du code d'un **essai** du score

```
mm(Essai,Code,[Bp,Mp]):- bplace(Essai,Code,Ressai,Rcode,Bp),
                         mplace(Ressai,Rcode,Mp).
```

```
bplace([],[],[],[],[]).
bplace([U—P],[U|C],P1,C1,[i|S]):-
    bplace(P,C,P1,C1,S).
bplace([V—P],[W|C],[V|P1],[W|C1],S):-
    diff(V,W),bplace(P,C,P1,C1,S).
```

```
mplace([],C,[]):-!.
mplace([U|P],C,[i|S]):-
  del(U,C,C1),!,mplace(P,C1,S).
mplace([U|P],C,S):-
  not(member(U,C)),mplace(P,C,S).
```

Recherche du code d'un essai du **score**

```
codepossible([],_).
codepossible( [[P1,S1]|Ps],Co):-
    mm(P1,Co,S1),codepossible(Ps,Co).
```

```
score(Essai,Total):-
    code(Bcode),
    mm(Essai,Bcode,Total).
```

```
mm(Essai,Code,[Bp,Mp]):- bplace(Essai,Code,Ressai,Rcode,Bp),
                         mplace(Ressai,Rcode,Mp).
```

```
bplace([],[],[],[],[]).
bplace([U|P],[U|C],P1,C1,[i—S]):-
    bplace(P,C,P1,C1,S).
bplace([V|P],[W|C],[V|P1],[W|C1],S):-
    diff(V,W),bplace(P,C,P1,C1,S).
```

```
mplace([],C,[]):-!.
mplace([U|P],C,[i—S]):-
  del(U,C,C1),!,mplace(P,C1,S).
mplace([U|P],C,S):-
  not(member(U,C)),mplace(P,C,S).
```

Le même but Prolog intervient de manières diverses dans le programme suivant les instantiations des variables, correspondant au type de recherche effectuée. Les explications associées sont ainsi différentes.

Annexe C. Notion de point de vue

Recherche du code

Le but diff(V,W)
Si un code possible est connu, ce but assure un simple test : les couleurs V et W sont bien diférentes. Sinon, c'est un but générateur qui permet d'instancier W avec toutes les couleurs différentes de V.

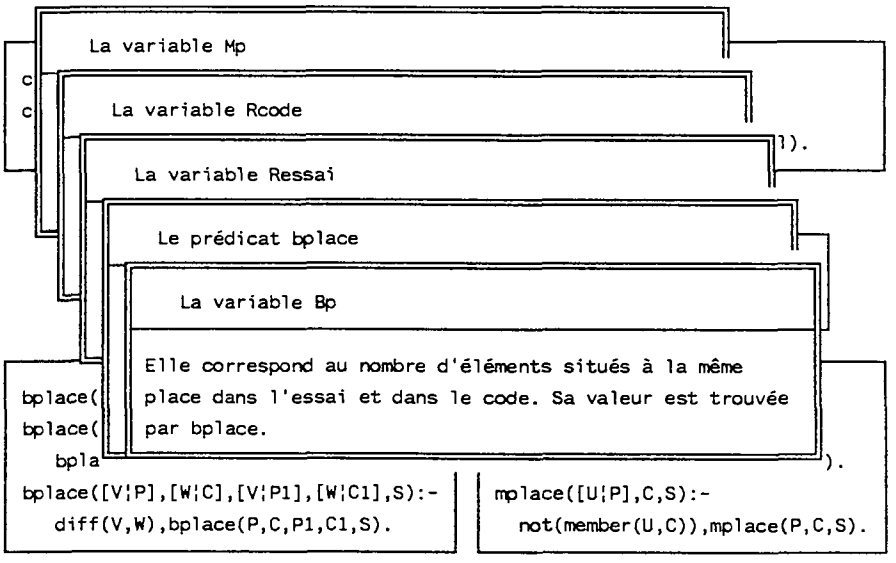
Recherche d'un essai

Le but diff(V,W)
C'est un but générateur qui permet d'instancier V avec toutes les couleurs différentes de W.

Recherche du score

Le but diff(V,W)
C'est un but testeur qui permet d'assurer que les couleurs V et W sont différentes.

L'hypertexte permet aussi une exploration complète du programme



C.a.2 Le programme MASTERMIND

A titre indicatif, voici le listage du programme MASTERMIND :

Lancement du programme :

```
go:-                    writes("Entrer le code :"),
                        read(Code),asserta(code(Code)),
                        nl,genere_code(P),
                        affiche_score(P,S,1),
                        etend([[P,S]],2),!,retract(code(_)).
```

Initialisations :

```
nbplace(4).
nbcouleur(6).
couleur(1,noir).
couleur(2,bleu).
couleur(3,vert).
couleur(4,rouge).
couleur(5,blanc).
couleur(6,jaune).
```

Affichage du score :

```
affiche_score(Cc,[Sp,Sm],Ns):-nl,
                  writes("Essai numéro "),display(Ns),writes(" : "),
                  display(Cc),nl,
                  score(Cc,[Sp,Sm]),
                  writes(" ---> "),
                  writes("bien placé : "),compte(Sp,Sp1),
                  display(Sp1),writes(" "),
                  writes("mal placé : "),compte(Sm,Sm1),
                  display(Sm1),nl.
```

Génération du code à trouver :

```
genere_code(X):-nbplace(N),gencode(X,N).

gencode([],0):-!.
gencode([U|V],N):-has(U),arith(sub,N,1,N1),gencode(V,N1).

has(X):-nbcouleur(Nb),randomax(N,Nb),couleur(N,X).

randomax(X,N):-random(Y),Z is Y mod N,X is Z + 1.
```

Recherche code, essai, score :

```
codepossible([],_).
codepossible([[P1,S1]|Ps],P):-mm(P1,P,S1),codepossible(Ps,P).
```

Annexe C. Notion de point de vue

```
score(Essai,Score):- code(C),mm(Essai,C,Score).


 mm(Essai,Code,[Bp,Mp]):-
          bplace(Essai,Code,Ressai,Rcode,Bp),
          mplace(Ressai,Rcode,Mp).

bplace([],[],[],[],[]).
bplace([U|P],[U|C],P1,C1,[i|S]):-bplace(P,C,P1,C1,S).
bplace([U|P],[V|C],[U|P1],[V|C1],S):-diff(U,V),bplace(P,C,P1,C1,S).


 mplace([],C,[]):-!.
 mplace([U|P],C,[i|S]):-del(U,C,C1),!,mplace(P,C1,S).
 mplace([U|P],C,S):-
          not(member(U,C)),
          mplace(P,C,S).
```

Divers :
(On peut supprimer la partie `tuple` ainsi que le premier test de couleur dans `diff`, sachant que le premier argument est toujours instancié)

```
 etend([[P,[N,_]]|_],_):-
          compte(N,Nb),nbplace(Nb),
          writes("Le code doit être :"),
          display(P),!,nl.

 etend(Cs,Ns):-
          codepossible(Cs,Cc),
          affiche_score(Cc,S,Ns),
          N is Ns + 1,
          etend([[Cc,S]|Cs],N).

compte([],0).
compte([_|X],N):-compte(X,N1),N is N1 + 1.

del(U,[U|Y],Y):-!.
del(U,[V|Y],[V|Y1]):-del(U,Y,Y1).

diff(C1,C2):-couleur(_,C1),couleur(_,C2),neq(C1,C2).
```

C.b LYRE

En ce qui concerne les différentes lectures d'un texte poétique, les exemples suivants illustrent quelques points de vue standards.

Le point de vue **IMAGES**

LE DORMEUR DU VAL

C'est un **trou de verdure** où **chante une rivière**
Accrochant follement aux herbes des **haillons**
D'argent; où le soleil, de la **montagne fière**,
Luit : c'est un petit **val qui mousse de rayons**.

Un soldat jeune, bouche ouverte, tête nue,
Et la nuque baignant dans le frais cresson bleu,
Dort; il est étendu dans l'herbe, sous la nue,
Pâle dans son **lit vert** où la **lumière pleut**.

Les **pieds dans les glaïeuls**, il dort. Souriant comme
Sourirait un enfant malade, il fait un somme:
Nature, berce-le chaudement: il a froid.

Les parfums ne font pas **frissonner** sa narine;
Il **dort dans le soleil**, la main sur sa poitrine
Tranquille. Il a deux trous rouges au côté droit.

Arthur RIMBAUD

Le point de vue **REJET**

LE DORMEUR DU VAL

C'est un trou de verdure où chante une rivière
Accrochant follement aux herbes des haillons
D'argent; où le soleil, de la montagne fière,
Luit : c'est un petit val qui mousse de rayons.

Un soldat jeune, bouche ouverte, tête nue,
Et la nuque baignant dans le frais cresson bleu,
Dort; il est étendu dans l'herbe, sous la nue,
Pâle dans son lit vert où la lumière pleut.

Les pieds dans les glaïeuls, il dort. Souriant comme
Sourirait un enfant malade, il fait un somme:
Nature, berce-le chaudement: il a froid.

Les parfums ne font pas frissonner sa narine;
Il dort dans le soleil, la main sur sa poitrine
Tranquille. Il a deux trous rouges au côté droit.

Arthur RIMBAUD

Le point de vue **CESURE**

LE DORMEUR DU VAL

C'est un trou de verdure où chante une rivière
Accrochant follement aux herbes des haillons
D'argent; où le soleil, de la montagne fière,
Luit : c'est un petit val qui mousse de rayons.

Un soldat jeune, **bouche** ouverte, tête nue,
Et la nuque baignant dans le frais cresson bleu,
Dort; il est étendu dans l'herbe, sous la nue,
Pâle dans son lit vert où la lumière pleut.

Les pieds dans les glaïeuls, il dort. Souriant comme
Sourirait un **enfant malade**, il fait un somme:
Nature, **berce-le** chaudement: il a froid.

Les parfums **ne font pas frissonner** sa narine;
Il dort dans le soleil, la main sur sa poitrine
Tranquille. Il a deux **trous rouges** au côté droit.

Arthur RIMBAUD

Le point de vue **GLOBAL** :
toutes les expressions expliquées suivant un point de vue quelconque sont mises en surbrillance. On peut choisir l'explication liée à un point de vue particulier :

LE DORMEUR DU VAL

C'est un **trou de verdure** où **chante une rivière**
Accrochant follement aux **herbes** des **haillons**
D'argent; où le **soleil**, de la **montagne fière**,
Luit : c'est un petit **val qui mousse de rayons**.

Un s verte, **tête** nue,
Et l le **frais cresson** bleu,
Dort 'herbe, sous la **nue**,
Pâle la **lumière pleut**.

Les s, il **dort**. Souriant comme
Sour , il fait un **somme**:
Natu t: il a **froid**.

Point de vue :
HOMME
NON-MOUVEMENT
REJET
SYNTAXE

Les **parfums ne font pas frissonner** sa **narine**;
Il **dort dans le soleil**, la **main** sur sa **poitrine**
Tranquille. Il a deux **trous rouges** au **côté** droit.

Arthur RIMBAUD

Annexe C. Notion de point de vue

Croisements :
exemple HOMME - NATURE

Les croisements permettent la mise en évidence simultanée de deux points de vue. Dans cet exemple, l'intersection entre les réseaux lexicaux HOMME et NATURE correspond aux expressions employées dans un sens inhabituel, ce qui est préparatoire aux IMAGES.

LE DORMEUR DU VAL

C'est un trou de verdure où **chante** une rivière
Accrochant follement aux herbes des **haillons**
D'argent; où le soleil, de la montagne **fière**,
Luit : c'est un petit val qui mousse de rayons.

Un **soldat** jeune, **bouche** ouverte, **tête** nue,
Et la **nuque** baignant dans le frais cresson bleu,
Dort; il est **étendu** dans l'herbe, sous la nue,
Pâle dans son **lit** vert où la lumière pleut.

Les **pieds** dans les glaïeuls, il **dort**. Souriant comme
Sourirait un **enfant malade**, il fait un **somme**:
Nature, berce-le chaudement: il a **froid**.

Les parfums ne font pas **frissonner** sa **narine**;
Il **dort** dans le soleil, la **main** sur sa **poitrine**
Tranquille. Il a deux trous rouges au **côté** droit.

Arthur RIMBAUD

C.c Un énoncé de problème de mathématiques

C'est un exemple d'utilisation de la notion de point de vue dans un problème de mathématiques élémentaires : la dynamique apparente de l'énoncé permet de mettre en évidence les éléments pertinents pour résoudre les différentes questions. Ce mode de visualisation peut être contrôlé par un module de diagnostic. En particulier, pour la question 3, deux possibilités sont offertes, suivant qu'une bonne réponse ait été fournie à la question 1 ou à la question 2.

Un train part de Paris pour aller à Lyon en passant par Dijon. Ce train démarre à 15h25 et s'arrête à Dijon 1h40 plus tard. Il y a 10mn d'arrêt à la gare de Dijon, puis il roule pendant 1h05 avant d'arriver à Lyon.

1) A quelle heure arrive le train à Dijon? ___
2) Quelle est la durée totale du trajet? ___
3) A quelle heure arrive le train à Lyon? ___

QUESTION 1

Un train part de Paris pour aller à Lyon en passant par Dijon. Ce train démarre à **15h25** et s'arrête à Dijon **1h40** plus tard. Il y a 10mn d'arrêt à la gare de Dijon, puis il roule pendant 1h05 avant d'arriver à Lyon.

QUESTION 2

Un train part de Paris pour aller à Lyon en passant par Dijon. Ce train démarre à 15h25 et s'arrête à Dijon **1h40** plus tard. Il y a **10mn** d'arrêt à la gare de Dijon, puis il roule pendant **1h05** avant d'arriver à Lyon.

QUESTION 3 a

Un train part de Paris pour aller à Lyon en passant par Dijon. Ce train démarre à **15h25** et s'arrête à Dijon 1h40 plus tard. Il y a 10mn d'arrêt à la gare de Dijon, puis il roule pendant 1h05 avant d'arriver à Lyon.

1) A quelle heure arrive le train à Dijon? ___
2) **Quelle est la durée totale du trajet** ? 2h55
3) A quelle heure arrive le train à Lyon? ___

QUESTION 3 b

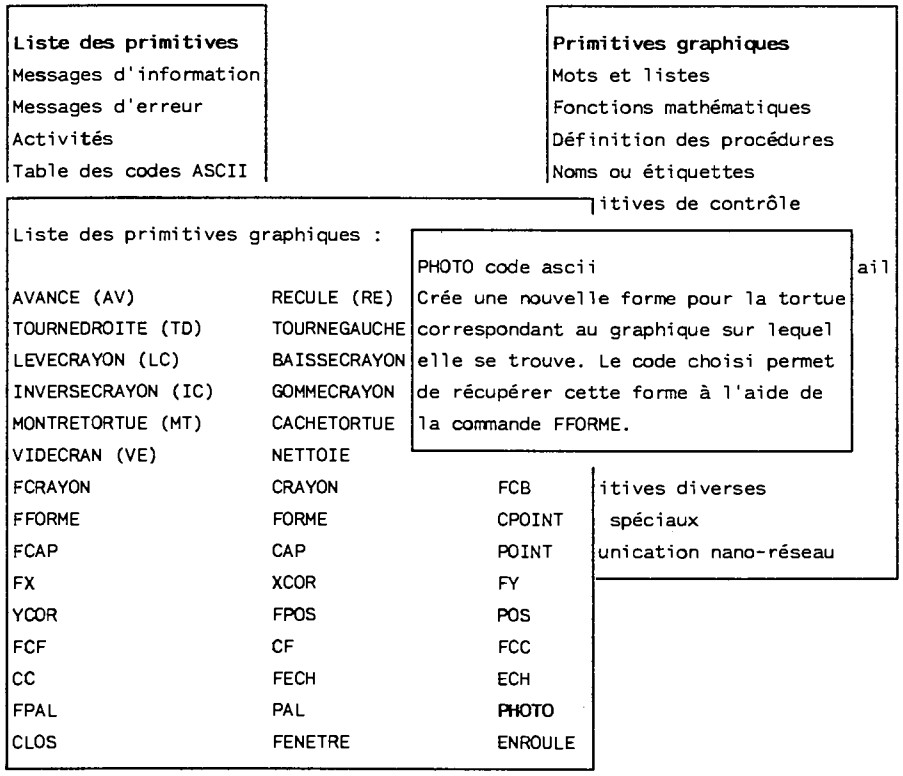
Un train part de Paris pour aller à Lyon en passant par Dijon. Ce train démarre à 15h25 et s'arrête à Dijon 1h40 plus tard. Il y a **10mn** d'arrêt à la gare de Dijon, puis il roule pendant **1h05** avant d'arriver à Lyon.

1) **A quelle heure arrive le train à Dijon** ? 17h05
2) Quelle est la durée totale du trajet? ___
3) A quelle heure arrive le train à Lyon? ___

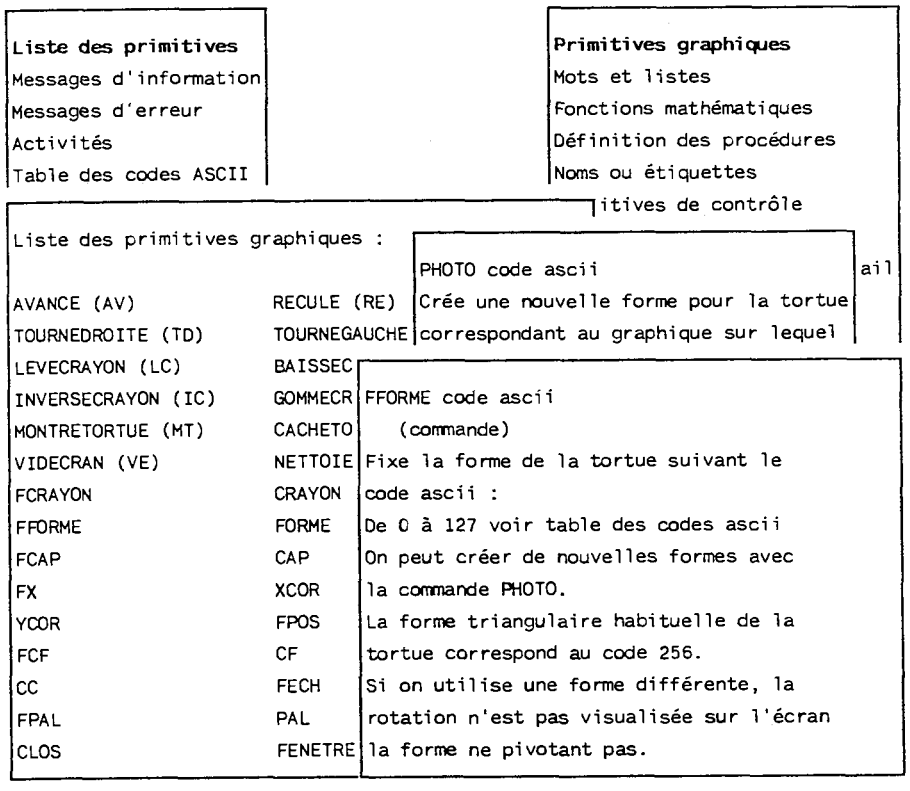
C.d Hyperinfo LOGO

Cette section montre différents types d'appel à l'aide en ligne du langage LOGO (environ 130K de code). Recherche arborescente pour trouver l'explication d'une primitive quelconque :

- appel du menu général,
- liste des primitives,
  - primitives graphiques,
    * PHOTO


Liste des primitives
Messages d'information
Messages d'erreur
Activités
Table des codes ASCII
Primitives graphiques
Mots et listes
Fonctions mathématiques
Définition des procédures
Noms ou étiquettes
itives de contrôle
Liste des primitives graphiques :
PHOTO code ascii
Crée une nouvelle forme pour la tortue correspondant au graphique sur lequel elle se trouve. Le code choisi permet de récupérer cette forme à l'aide de la commande FFORME.
ail
AVANCE (AV)
RECULE (RE)
TOURNEDROITE (TD)
TOURNEGAUCHE
LEVECRAYON (LC)
BAISSECRAYON
INVERSECRAYON (IC)
GOMMECRAYON
MONTRETORTUE (MT)
CACHETORTUE
VIDECRAN (VE)
NETTOIE
FCRAYON
CRAYON
FCB
itives diverses
FFORME
FORME
CPOINT
spéciaux
FCAP
CAP
POINT
unication nano-réseau
FX
XCOR
FY
YCOR
FPOS
POS
FCF
CF
FCC
CC
FECH
ECH
FPAL
PAL
PHOTO
CLOS
FENETRE
ENROULE

A partir de la fiche PHOTO, appel de la fiche liée à une primitive corrélée FFORME. Un bouclage est ici possible, puisque la fiche FFORME permet à son tour de rappeler la fiche PHOTO. L'utilisateur est tout à fait libre de tourner en rond s'il le désire.



Exemple de recherche débouchant sur une fiche descriptive d'une primitive à partir d'un choix d'activité :

```
Liste des primitives
Messages d'information
Messages d'erreur
Activités
Table des codes ASCII
Touches fonction
```

```
Comment faire pour :
...
Définir une procédure
Définir une propriété
Redéfinir une primitive
Corriger une procédure
Sauver
Ramener
Effacer
Grouper
Imprimer
```

```
DEFINIS nom liste
    (commande)
Permet de définir dynamiquement
une procédure en faisant de "liste"
la définition de "nom".
Cette "liste" est une liste de listes :
  - la première liste contient la suite
    des noms d'arguments non précédés
    du caractère : ;
  - les autres éléments correspondent
    chacun à une ligne de la définition
    de la procédure.
Voir COPIEDEF et TEXTE.
```

```
Pour définir une procédure :
- taper POUR dans l'espace de
travail ;
- aller dans l'éditeur ;
- utiliser DEFINIS.
```

Accès direct à une fiche à partir de son nom ou l'un de ses synonymes d'accès. Continuation de la navigation hypertexte à partir de cette racine.

```
?FLECTURE "TOTO.LOG
```

```
FLECTURE  port
FLECTURE  nomfich
       (commande)
Fixe la source de lecture
(l' entrée courante) à "port"
ou à "nomfich". Avant d'utiliser
cette commande, il faut ouvrir le
fichier ou le port avec OUVRE.
FLECTURE fixe la position de lecture
au début du fichier (voir FPOSLEC).
On revient à la lecture au clavier
en tapant FLECTURE "CON (ou []).
Voir FDFLEC?.
```

```
OUVRE port
OUVRE nomfich
      (commande)
Ouvre le périphérique port ou le
fichier nomfich (avec son extension),
pour les transformer en sortie courante
(écriture), ou en entrée courante
(lecture). Si le fichier "nomfich"
n'existe pas, cette commande crée
automatiquement un fichier avec ce nom.
Voir FERME et FERMETOUT.
```

C.e Exemple d'utilisation d'HyperInfo en grammaire

Cet exemple a pour but de montrer brièvement les liens que l'on peut tisser entre un texte quelconque et une grammaire organisée en fiches hypertextes.

Tout d'abord, un exemple de texte truffé de fautes d'orthographes. Vous pouvez essayer mentalement d'en déterminer le nombre.

REVERIE CHAMPETRE

Vêtu d'un de leurs pull-over bleu, au son des tournes-disques, elles se sont baigné dans la rivière, toutes étonnées, puis se sont lavées les mains en négligent les avants-derniers oiseaux-mouche roses pâle prêt des arc-en-ciel. Elles pensaient a rentré classer leurs timbre-poste rose à la lueur des abat-jours oranges. Quand aux garde-chasse négligeant, ils étaient venu plutôt, tous seuls, après avoir chausser leur botte. En mangeant les pâtisseries aigre-douces des grands-mères pendant des demies-journées, ils comptaient les pattes dans les basse-cours : quatre-vingt, quatre-vingt-cinq, deux cent, deux cent-cinquante, voir deux milles.

Nombre de fautes trouvées :

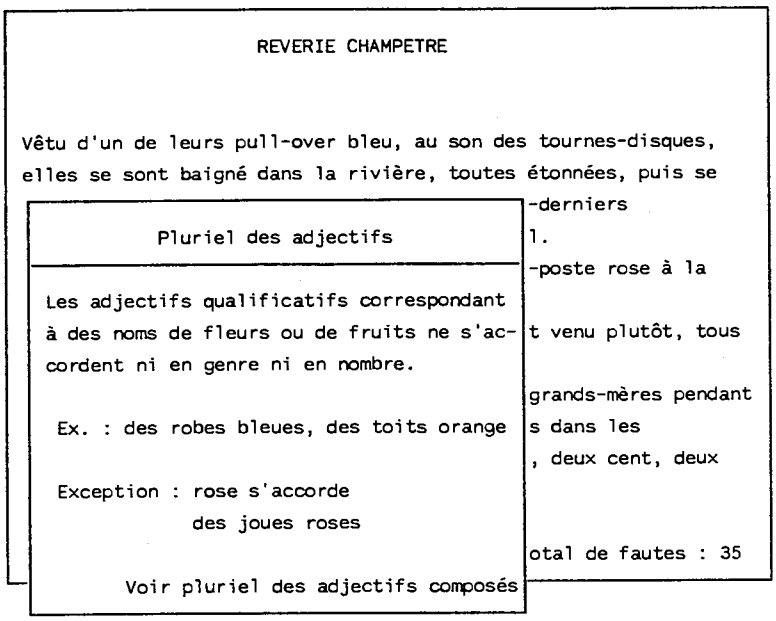
La correction s'effectue en associant à chaque mot mal orthographié, la règle de grammaire qui lui correspond.

REVERIE CHAMPETRE

Vêtu d'un de leurs **pull-over bleu**, au son des **tournes-disques**, elles se sont **baigné** dans la rivière, **toutes** étonnées, puis se sont **lavées** les mains en **négligent** les **avants-derniers oiseaux-mouche roses pâle prêt** des **arc-en-ciel**. Elles pensaient **a rentré** classer leurs **timbre-poste rose** à la lueur des **abat-jours oranges**. **Quand** aux **garde-chasse négligeant**, ils étaient **venu plutôt**, **tous** seuls, après avoir **chausser leur botte**. En mangeant les pâtisseries **aigre-douces** des **grands-mères** pendant des **demies-journées**, ils comptaient les pattes dans les **basse-cours : quatre-vingt**, quatre-vingt-cinq, deux **cent**, deux cent-cinquante, **voir** deux **milles**.

Nombre total de fautes : 35

Comme dans les exemples précédents, il suffit de se positionner sur le mot choisi et de cliquer pour faire apparaître la règle incriminée.



Les règles incluses correspondent à la hiérarchie suivante :

Quelques éléments de grammaire

Accord du participe passé	Etude de quelques homophones	Pluriel des mots composés
Employé comme épithète Avec l'auxiliaire être Avec l'auxiliaire avoir Forme pronominale	Des homophones sont des mots qui se prononcent de la même façon et s'écrivent différemment : Quelques exemples : a ou à quand ou quant plus tôt ou plutôt prêt ou près voir ou voire	Il dépend du sens de chaque mot composé. On peut donner cependant certaines règles liées aux types de constituants : Nom + nom Nom + préposition + nom Adjectif + nom Adjectif + adjectif Verbe + nom Mot invariable + nom ou adjectif Verbe + verbe Mots étrangers Voir aussi l' adjectif numéral cardinal

Les liaisons contextuelles ne peuvent s'effectuer directement à partir d'un mot quelconque de la langue française (sauf pour les règles ad hoc sur les homophones). Un système automatique nécessiterait une analyse grammaticale complète.

L'idée est ici simplement de permettre la déclaration de synonymes, i.e. des nouveaux modes d'accès aux règles déjà constituées. Ainsi, à la règle d'accord du pluriel des adjectifs composés, on ajoute les synonymes : "AIGRE-DOUCES", "AVANTS-DERNIERS", "ROSES PALE"; pour le pluriel des adjectifs, on ajoute les adjectifs de couleur : "BLEU", "ORANGES", "ROSE".

C'est une façon simple de permettre l'adaptation d'un logiciel à une situation particulière, qui ne nécessite pas le recours à des outils très sophistiqués.

C.f L'application PHILO

Ce logiciel d'explication de textes philosophiques a été réalisée avec le concours de Jean d'Yvoire. Il se compose de cinq textes expliqués suivant une méthodologie générale. On prendra l'exemple d'un texte de Spinoza pour l'illuster.

> *« Pour ma part, je dis que cette chose est libre qui existe et agit par la seule nécessité de sa nature, et contrainte cette chose qui est déterminée par une autre à exister et à agir selon une modalité précise et déterminée. Dieu, par exemple, existe librement (quoique nécessairement) parce qu'il existe par la seule nécessité de sa nature. De même encore, Dieu connaît soi-même et toutes choses en toute liberté parce qu'il découle de la seule nécessité de sa nature qu'il comprenne toutes choses. Vous voyez donc que je ne situe pas la liberté dans un libre décret, mais dans une libre nécessité. Mais venons-en aux choses créées qui, toutes, sont déterminées à exister et à agir selon une manière précise et déterminée. Pour le comprendre clairement, prenons un exemple très simple. Une pierre reçoit d'une cause extérieure qui la pousse une certaine quantité de mouvement par laquelle elle continuera nécessairement de se mouvoir après l'arrêt de l'impulsion externe. Cette permanence de la pierre dans son mouvement est une contrainte, non pas parce qu'elle est nécessaire, mais parce qu'elle doit être définie par l'impulsion des causes externes; et ce qui est vrai de la pierre, l'est aussi de tout objet singulier, quelle qu'en soit la complexité et quel que soit le nombre de ses possibilités : tout objet singulier, en effet, est nécessairement déterminé par quelque cause extérieure à exister et à agir selon une loi précise et déterminée.*
> *Concevez maintenant, si vous voulez bien, que la pierre, tandis qu'elle continue de se mouvoir, sache et pense qu'elle fait tout l'effort possible pour continuer de se mouvoir. Cette pierre, assurément puisqu'elle n'est consciente que de son effort, et qu'elle n'est pas indifférente, croira être libre et ne persévérer dans son mouvement que par la seule raison qu'elle le désire.Telle est cette liberté humaine que tous les hommes se vantent d'avoir et qui consiste en cela seul que les hommes sont conscients de leurs désirs et ignorants des causes qui les déterminent. C'est ainsi qu'un enfant croit désirer librement le lait, et un jeune garçon irrité vouloir se venger s'il est irrité, mais fuir s'il est craintif. Un ivrogne croit dire par une décision libre ce qu'ensuite il aurait voulu taire. De même un dément, un bavard, et de nombreux cas de ce genre croient agir par une libre décision de leur esprit, et non portés par une impulsion. Et, comme ce préjugé est inné en tous les hommes, ils ne s'en libèrent pas facilement. »*
>
> Spinoza : Lettre à Schuller

Voici l'architecture générale de l'analyse des textes, incluant certains éléments spécifiques du texte de Spinoza.

La méthode se décompose en quatre analyses distinctes :

1. Thèmes
2. Thèses
3. Articulations
4. Problématique

THEMES		
Méthodologie	Thèmes principaux	Croisements
Explication de la méthode	1- Liberté et nécessité 2- Les êtres dont parle Spinoza 3- Mouvement et volonté 4- La connaissance 1- Liberté et nécessité 1.1- La liberté 1.2- La contrainte 1.3- La liberté humaine	Choix d'un sous-thème 1- Liberté et nécessité 1.1- La liberté 1.2- La contrainte 1.3- La liberté humaine 2- Les êtres dont parle Spinoza 2.1- Dieu 2.2- Les choses créées 2.3- L'homme 3- Mouvement et volonté 3.1- Mouvement 3.2- Désir et volonté 4- La connaissance 4.1- Connaissance et ignorance 4.2- Conscience 4.3- Croyance

THESES		
Méthodologie	Thèses du texte	Croisements
Explication de la méthode	1- Prémisses 2- Conclusions	Thème 1 Thème 2

ARTICULATIONS			
Méthodologie	Toutes les articulations	Certaines articulations	Croisements
Explication de la méthode	Affichage complet	Opposition Induction Hypothèse Explication Exemplification Déduction Alternative Conjonction Concession Comparaison	Thèmes 1 et 2 Thèses

PROBLEMATIQUE		
Méthodologie	Première approche	Approfondissement

C.f. L'application PHILO

C.g L'application AIP

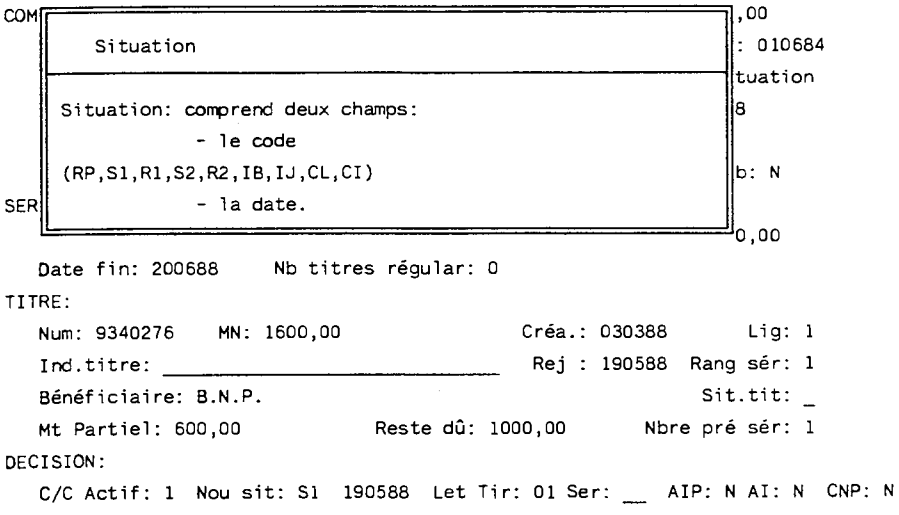
Ce logiciel sur l'apprentissage du traitement informatisé des incidents de paiement se compose d'une simulation de l'application réelle, de divers cas et d'un générateur de cas. Une couche hypertexte permet d'expliciter tous les termes apparaissant sur les écrans de saisie.

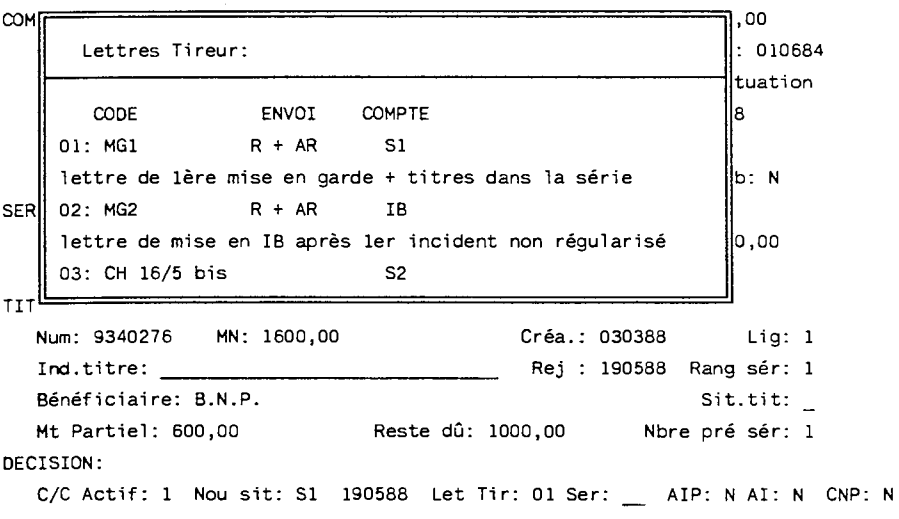
Exemples d'écrans :

```
DGP / FIP            SAISIE COMPLEMENTS - VALIDATION-ACTION     P 3.1   20/05/88
                              SAISIE COMPLEMENTS                  2/3
--------------------------------------------------------------------------------

COMPTE:  4 986 71
   Nom: ETOURNEAU PAUL
   Indic  sp\'eciale 1: COMPTE TITRE
   Indic  sp\'eciale 2:
   Indic  sp\'eciale 3:


TITRE:
   Num: 9340276    Mt: 1600,00              Cr\'eat:  ______
   Ind titre:                                Rej :  190588
   B\'en\'eficiaire: B.N.P._____________________
   Mt.Partiel: 60000_______

--------------------------------------------------------------------------------


DGP / FIP            SAISIE COMPLEMENTS - VALIDATION-ACTION     P 3.1   20/05/88
                                VALIDATION-ACTION                 3/3
--------------------------------------------------------------------------------
COMPTE:  4 986 71    Part: P1=00 P2=00 P3=00 P4=00    Mt oppos: 0,00
   Nom: ETOURNEAU PAUL                               Date Ouvert: 010684
   Indic. sp\'eciale 1: COMPTE TITRE_____________     Ancienne situation
   Indic. sp\'eciale 2: _________________________        RP 190588
   Indic. sp\'eciale 3: _________________________
   Avoir c/c: -768,41        D\'ecaut: 500,00          C/C r\'eattrib: N
SERIE:
   Date d\'eb: 190588     Nbre titres s\'erie: 1      Mt Impay\'es: 1000,00
   Date fin: 200688     Nb titres r\'egular: 0
TITRE:
   Num: 9340276    MN: 1600,00               Cr\'ea.: 030388       Lig: 1
   Ind.titre: _____________________________   Rej : 190588  Rang s\'er: 1
   B\'en\'eficiaire: B.N.P.                                     Sit.tit: _
   Mt Partiel: 600,00           Reste d\^u: 1000,00       Nbre pr\'e s\'er: 1
DECISION:
   C/C Actif: 1  Nou sit: S1  190588  Let Tir: 01 Ser: __  AIP: N AI: N  CNP: N
```

Exemples de lancement de l'hypertexte :

On peut remarquer que le fenêtrage permet de bien différencier ce qui appartient à l'application réelle de ce qui est ajouté dans la version d'apprentissage, par effet de superposition.





On peut alors explorer le sens des divers codes de situation ou lire les lettres envoyées au titulaire du compte :

C.C.P PARIS — Paris, le

Téléphone

Postes

Références (N° Rec.) — INTERDICTION D'EMETTRE DES CHEQUES

Ch 16/14

M ,

Conformément à la loi, je vous confirme la mesure d'interdiction bancaire qui vous a été notifiée par la lettre recommandée le .

La faculté d'émettre des chèques vous est retirée jusqu'au .

En effet, je suis au regret de constater que vous n'avez pas régularisé l'incident de paiement survenu sur votre compte durant le délai légal de 30 jours.

Toutefois dans le cas...

Annexe C. Notion de point de vue

Annexe D

Le système de vérification de la démonstration dans ARRIA

D.a Exemples d'exercice d'ARRIA

D.a.1 Exercice I

Enoncé :

Soient 4 points A, B, C, D tels que $\vec{AB} = \vec{DC}$ et $||\vec{AB}|| = ||\vec{AD}||$.
Montrer que les vecteurs $\vec{AC}$ et $\vec{BD}$ sont orthogonaux.

Liste des fragments :

```
frag(1,h,"||AB|| = ||AD||").
frag(2,r,"les droites (AC) et (BD) sont perpendiculaires").
frag(3,h,"on sait que AB = DC").
frag(4,o,"les diagonales d'un losange sont perpendiculaires").
frag(5,o,"(A,B,C,D) est un parallélogramme si et seulement si AB = DC").
frag(6,o,"un parallélogramme ayant 2 côtés consécutifs égaux est un
losange").
frag(7,r,"(A,B,C,D) est un parallélogramme").
frag(8,r,"(A,B,C,D) est un losange").
frag(9,c,"les vecteurs AC et BD sont orthogonaux").
```

Arbre :

```
prouve(2,[4,8]).
prouve(7,[3,5]).
prouve(8,[1,7,6]).
prouve(9,[2]).
```

D.a.2 Exercice II

Enoncé :

Soient 2 triangles (A, B, C) et (M, N, P) ayant le même centre de gravité G.
Montrer que $A\vec{M} + B\vec{N} + C\vec{P} = \vec{0}$

Liste des fragments :

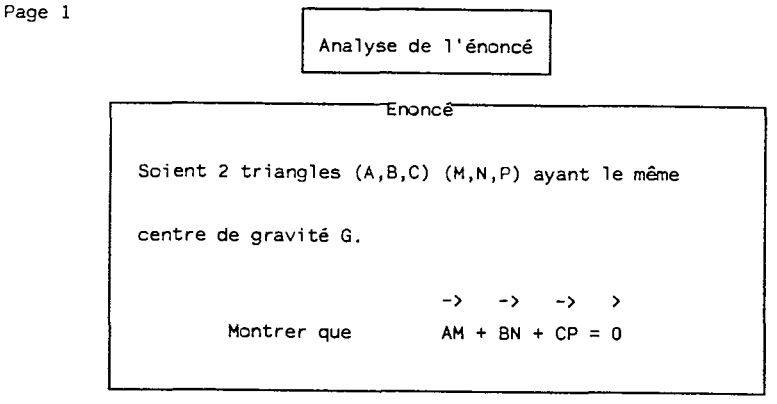
```
frag(1,r,"AG + BG + CG = 0").
frag(2,h,"G est le centre de gravité de (A,B,C)").
frag(3,r,"AM + BN + CP = (AG + BG + CG) + (GM + GN + GP)").
frag(4,o,"on connaît la relation de Chasles").
frag(5,r,"GM + GN + GP = 0").
frag(6,c,"on peut conclure que AM + BN + CP = 0").
frag(7,r,"AM + BN + CP = (AG + GM) + (BG + GN) + (CG + GP)").
frag(8,h,"G est le centre de gravité de (M,N,P)").
frag(9,r,"GA + GB + GC = 0").
```

Arbre :

```
prouve(1,[9]).
prouve(3,[7]).
prouve(5,[8]).
prouve(6,[1,5,3]).
prouve(7,[4]).
prouve(9,[2]).
```

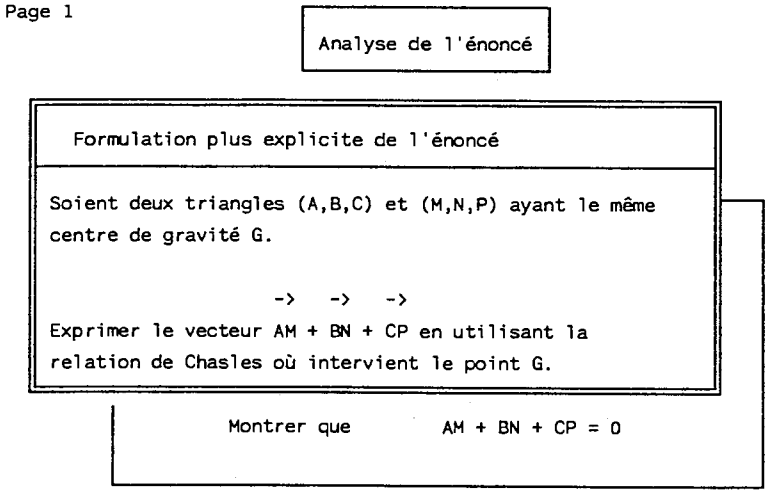
D.b Déroulement d'une session (exercice II)

Phase 1 : Lecture active de l'énoncé

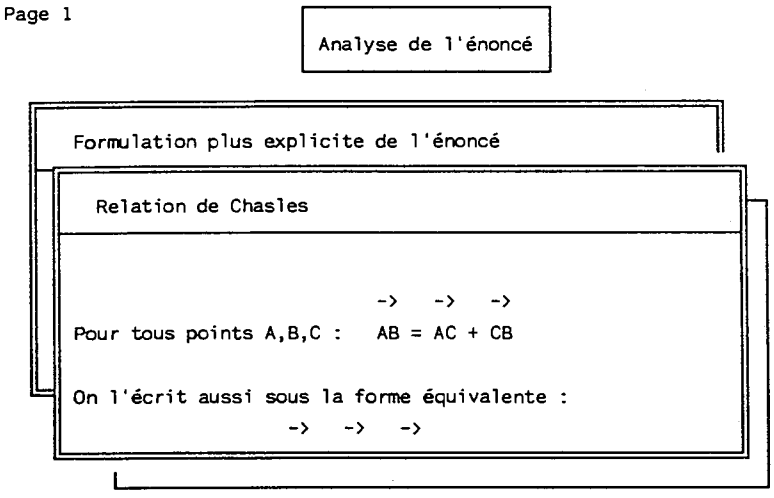


Annexe D. Le système de vérification de la démonstration dans ARRIA

Appel d'un énoncé plus stratégique

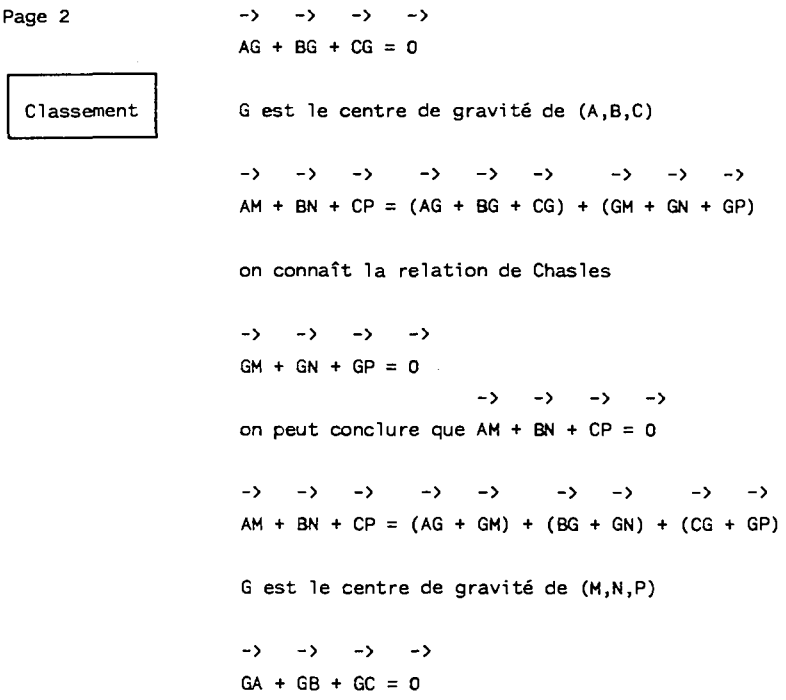


Recherche d'informations complémentaires

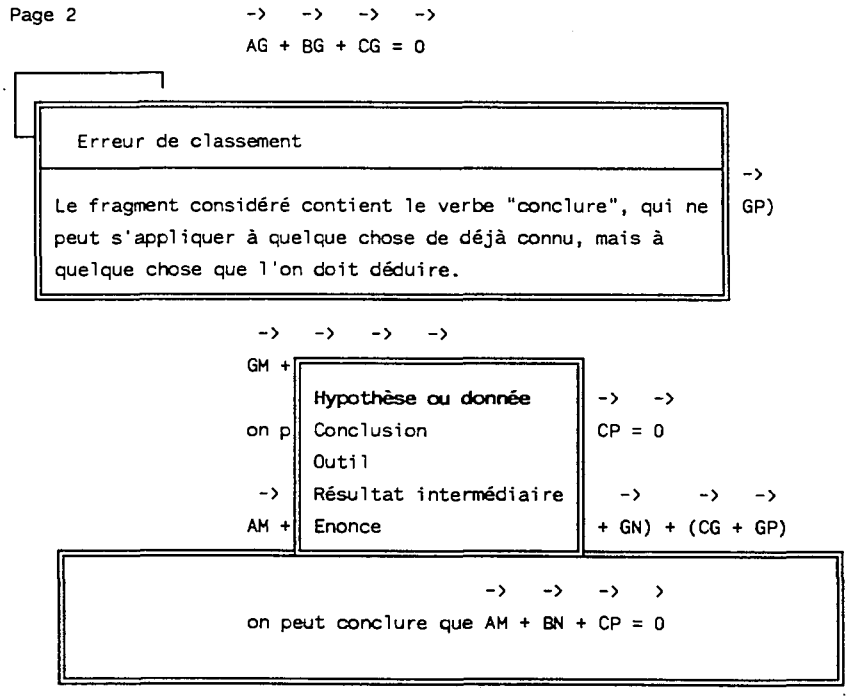


D.b. Déroulement d'une session (exercice II)

Phase 2 : Typage des fragments

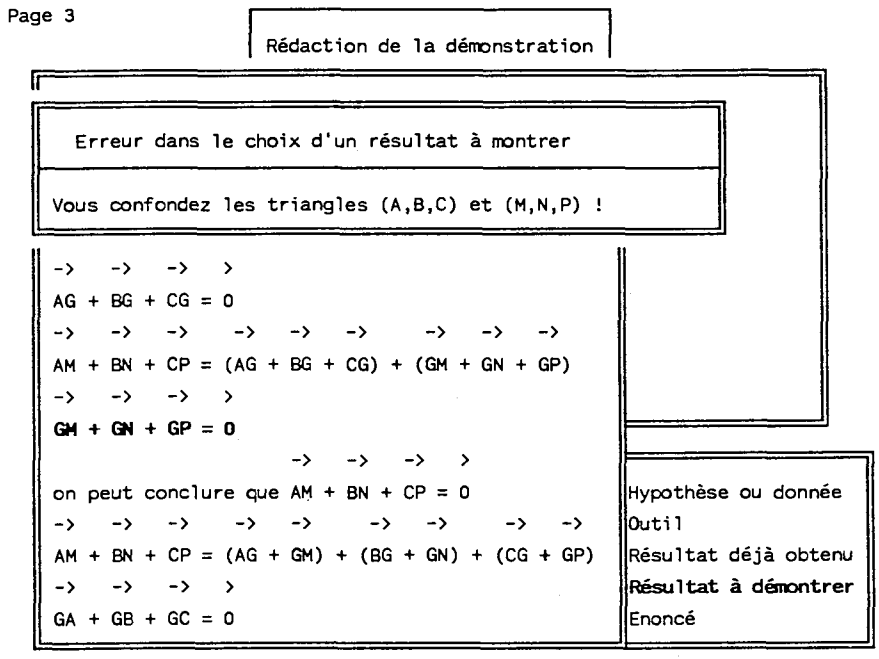


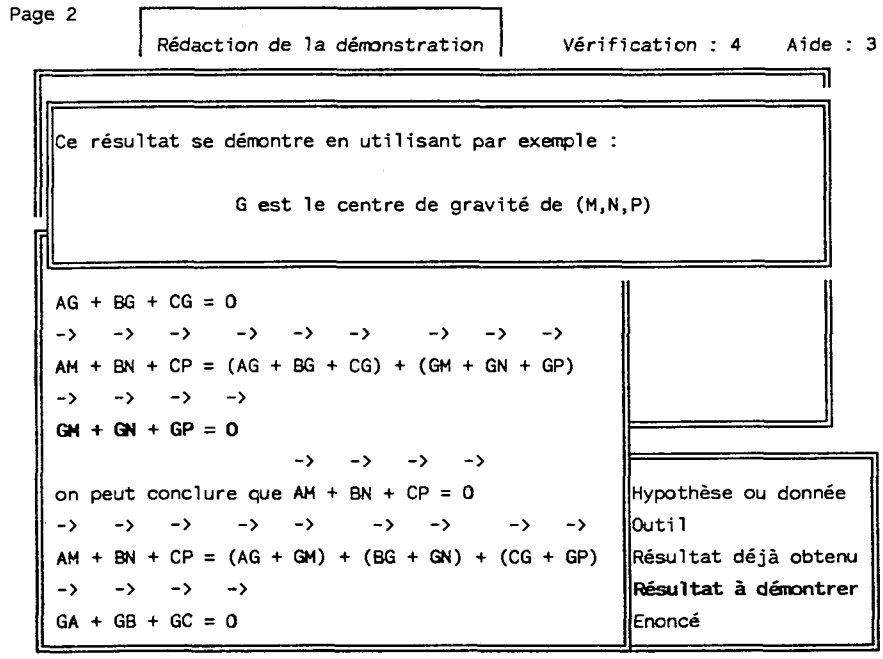
Exemple de message d'erreur



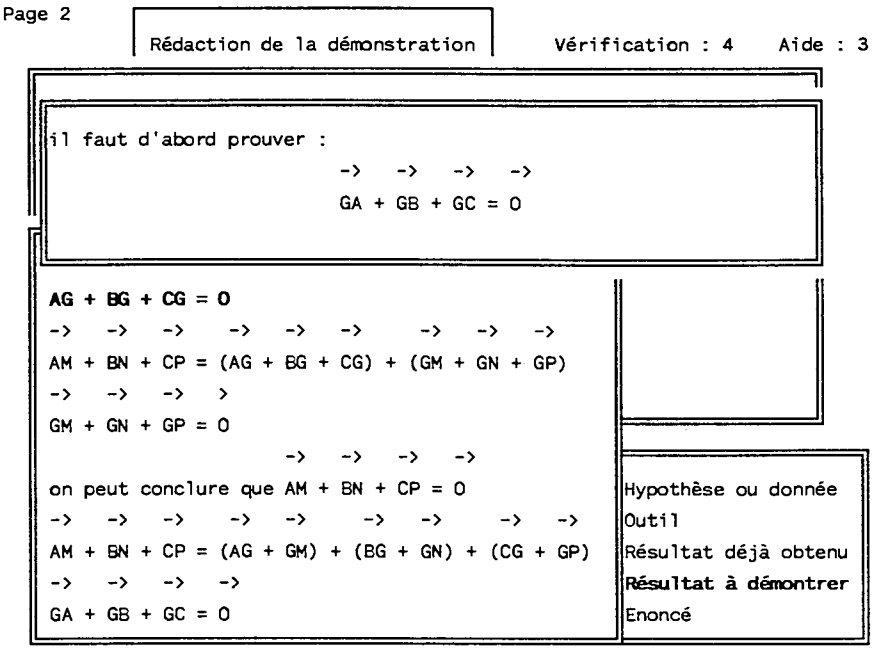
Phase 3 : Rédaction d'une démonstration

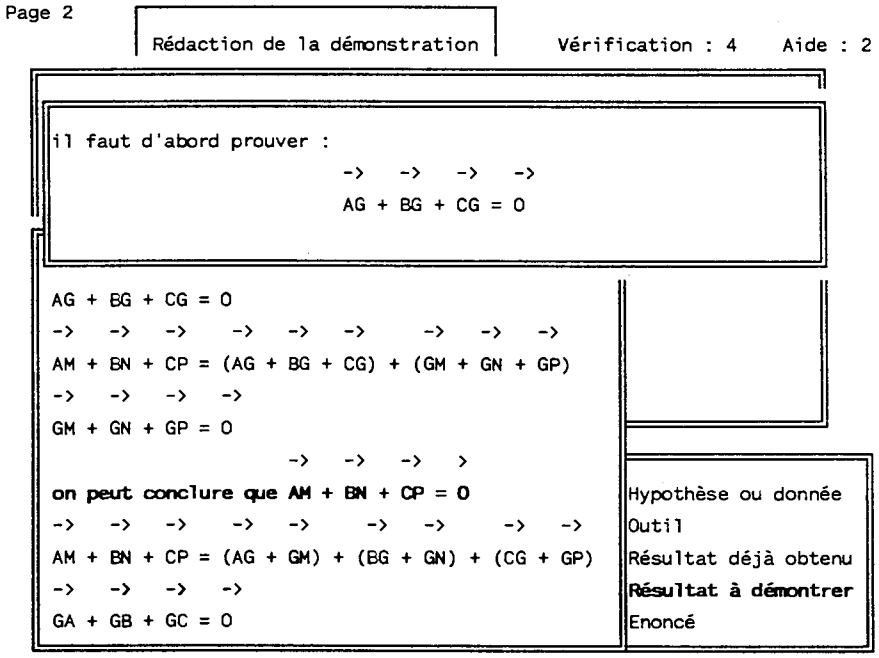
Quelques messages d'erreur, après avoir introduit l'hypothèse : G est le centre de gravité de (A, B, C).





D.b. Déroulement d'une session (exercice II)





Annexe D. Le système de vérification de la démonstration dans ARRIA

Exemples de démonstration

G est le centre de gravité de (A,B,C)

donc $\vec{GA} + \vec{GB} + \vec{GC} = \vec{0}$

ou $\vec{AG} + \vec{BG} + \vec{CG} = \vec{0}$.

car $\vec{GM} + \vec{GN} + \vec{GP} = \vec{0}$
G est le centre de gravité de (M,N,P).

On connaît la relation de Chasles

donc $\vec{AM} + \vec{BN} + \vec{CP} = (\vec{AG} + \vec{GM}) + (\vec{BG} + \vec{GN}) + (\vec{CG} + \vec{GP})$

donc $\vec{AM} + \vec{BN} + \vec{CP} = (\vec{AG} + \vec{BG} + \vec{CG}) + (\vec{GM} + \vec{GN} + \vec{GP})$

or $\vec{AG} + \vec{BG} + \vec{CG} = \vec{0}$

et $\vec{GM} + \vec{GN} + \vec{GP} = \vec{0}$

donc **on peut conclure que** $\vec{AM} + \vec{BN} + \vec{CP} = \vec{0}$.

DEMONSTRATION

On connaît la relation de Chasles

donc $\vec{AM} + \vec{BN} + \vec{CP} = (\vec{AG} + \vec{GM}) + (\vec{BG} + \vec{GN}) + (\vec{CG} + \vec{GP})$

ou $\vec{AM} + \vec{BN} + \vec{CP} = (\vec{AG} + \vec{BG} + \vec{CG}) + (\vec{GM} + \vec{GN} + \vec{GP})$.

car $\vec{GA} + \vec{GB} + \vec{GC} = \vec{0}$
G est le centre de gravité de (A,B,C)

donc $\vec{AG} + \vec{BG} + \vec{CG} = \vec{0}$.

G est le centre de gravité de (M,N,P)

donc $\vec{GM} + \vec{GN} + \vec{GP} = \vec{0}$.

On peut conclure que $\vec{AM} + \vec{BN} + \vec{CP} = \vec{0}$.

D.c Description générale

ARRIA se compose d'environ 45 K de code Prolog réparti dans plusieurs fichiers (problèmes d'overlay) :

- Répertoire principal

```
FPROLOG  INI      6905~: outils généraux
```

- Répertoire CREATION

```
VERIF    PRO     2057 : vérification des déclarations
SAUVER   PRO     1570 : sauvegarde des déclarations
CHARGER  PRO      990 : chargement d'un exercice
INIT     PRO       85 : initialisations
IMPRIME  PRO     1584 : impression d'un exercice
GENERAL  PRO     6520 : outils généraux pour la création (en mémoire)
CARAC    PRO      168 : caractères spéciaux intégrables
```

- Répertoire UTIL

```
INIT     PRO      583 : initialisations / lancement
AIDE     PRO      242 : changement du type d'aide en cours d'exercice
IMP      PRO      826 : impression durant le classement
AFFICHE  PRO     1059 : affichage de l'exercice courant
VERIFS   PRO     1945 : vérification en fin de bloc
IMPRIME  PRO     3206 : impression générale
GENERAL  PRO    11082 : outils généraux (en mémoire)
CLASSER  PRO     3152 : classement des fragments
DEMO     PRO     1101 : lancement de la démonstration
TYPES    PRO     1132 : changement aide/vérification
```

La description complète du programme semble superflu, le plus intéressant concerne la partie spécifique de la vérification en pas à pas de la démonstration, expliquée dans la suite.

D.d Vérification de la démonstration

La vérification de la démonstration dans ARRIA s'effectue en plusieurs étapes comprenant chacune plusieurs phases :

- vérification d'un bloc, i.e. démonstration d'un résultat,
- vérification des divers choix (connecteurs, fragments) à l'intérieur d'un bloc.

Cette dernière vérification se décompose en 4 moments distincts :

1. vérification d'un fragment en début de bloc,
2. vérification d'un fragment en cours de constitution d'un bloc,
3. vérification d'un connecteur en cours de constitution d'un bloc,
4. vérification d'un connecteur en fin de bloc.

La vérification d'un fragment en cours de bloc s'effectue en deux étapes :

- vérification du type de fragment choisi (syntaxe),
- vérification du fragment choisi dans le type sélectionné (sémantique).

D.d.1 Processus général

La vérification du processus de démonstration utilise les variables suivantes :

1. liste des résultats à montrer (les numéros de fragment),
2. liste de ce qui est déjà fait (le chemin),
3. bloc en cours de réalisation (uniquement les numéros de fragment),
4. Résultat courant à démontrer (cela peut être une liste),
5. les attentes éventuelles (ou nil)

Deux variables globales interviennent :

- type_verif qui spécifie le type de grammaire choisi,
- type_aide qui correspond au niveau d'aide.

Le choix d'un des types de vérification s'effectue suivant le type de choix (connecteur ou fragment) et à l'aide de deux tests :

- si le bloc courant est vide, c'est la vérification d'un fragment en début de bloc,
- le prédicat 'fin_de_bloc' qui décide si la démonstration du bloc élémentaire courant est terminé ou non.

```
/* fin-de-bloc (Liste des résultats non démontrés, Bloc,Résultat courant) */

fin-de-bloc(AD,Bloc,R):-eval(type_verif,N),eq(4,N), !,
                member(R,Bloc),prouve(R,L),
                inclus(Bloc,[R|L]),tous_connus(L,Bloc,AD).

fin_de_bloc(_,Bloc,R):- prouve(R,L),inclus([R|L],Bloc).

tous_connus([],_,_).
tous_connus([X|L],Bloc,AD):-member(X,Bloc), !,tous_connus(L,Bloc,AD).
tous_connus([X|L],_,AD):-member(X,AD), !,fail.
tous_connus([X|L],Bloc,AD):-enonce(X,r,_), !,tous_connus(L,Bloc,AD).
```

Les types de fragments en cours de démosntratiob sont notés 1, 2, 3 ou 4 :

- 1 : hypothèse
- 2 : outil
- 3 : résultat déjà démontré
- 4 : résultat à démontrer

Le prédicat 'demon' incrémente le compteur des erreurs et lance un message d'erreur adapté à la situation courante :

```
demon(N):-add_cont(nb_err),lancer_expli(N,"DEMON"),fail.
lancer_expli(M,Nom) :-  string_from(M,M1),concat(Nom,M1,M2),
                        bandeau(1),lan_expli(M2).
```

D.d.2 Vérification début de bloc

Le premier moment consiste dans le choix d'un type de fragment (qui n'est pas vérifié), suivi du choix du fragment.

`verif_init` a 5 paramètres :

- liste des résultats à démontrer,
- type de fragment choisi
- choix effectué (numéro de fragment),
- bloc courant après le choix,
- connecteur devant suivre le choix courant.

Vérification d'un résultat à montrer : on se contente de regarder s'il est prouvable, i.e. s'il existe une relation de preuve ne contenant aucun résultat non encore démontré.

```
verif_init(AD,4,Choix,[Choix],[car]):-prouvable(Choix,AD), !.
verif_init(AD,4,Choix,_,_):- !,eval(type_aide,TA),mess_prouve(Choix,AD,TA).

Vérification d'un autre type de fragment~: on regarde si ce fragment intervient
dans une preuve d'un résultat prouvable non encore démontré.

verif_init(AD,_,Choix,LP,nil):- peut_prouver(Choix,LP,AD),neq(LP,[]), !.
verif_init(_,_,_,_,_):-demon(3).
verif_init(AD,_,Choix,_,_):-~ !,eval(type_aide,TA),mess_pr(Choix,AD,TA).
```

Les prédicat `mess_pr` et `mess_prouve` fournissent une information complémentaire dépendant du niveau d'aide courant et des relations de preuve déclarées :

```
mess_prouve(C,AD,_):-    prouve(C,[X]),prouvable(X,AD), !,mess_err(X,1).
mess_prouve(C,AD,1):-    enonce(Choix,c,_), !,demon(4).
mess_prouve(C,AD,2):-    prouve(C,L),commun(L,AD,L1),
                         eq(L1,[X|_]), !,mess_err(X,1).
mess_prouve(_,AD,3):-    mem_gen(Y,AD),prouve(Y,L),disjoint(L,AD),
                        !,mess_err(Y,2).
mess_prouve(_,_,_):-demon(2).


mess_pr(Choix,AD,2):-    mem_gen(C,AD),prouve(C,L),member(Choix,L),
                         commun(L,AD,L1),eq(L1,[X|_]), !,mess_err(X,1).
mess_pr(Choix,AD,3):-   !,mess_prouve(_,AD,3).
```

D.d.3 Vérification en cours de bloc

`verif_cours` se contente de vérifier le type de fragment choisi. Il nécessite 5 paramètres :

- type de fragment choisi
- bloc courant avant le choix,
- liste des résultats pouvant être démontrées à partir des éléments du bloc courant,
- attente d'un type de fragment (exemple attente d'un résultat à montrer après un 'donc'),

- **connecteur devant suivre le choix courant.**

```
verif_cours(4,_,_,[4],[car]):- !.
verif_cours(4,Bloc,[R],_,_):- member(R,Bloc), !,demon(12).
verif_cours(4,_,_,_,_):-~ !,demon(13).
verif_cours(_,_,_,[4],_):-~ !,demon(14).
verif_cours(X,_,_,L,nil):-member(X,L), !.
verif_cours(_,_,_,_,_):-demon(15).
```

L'acceptation d'un type de fragment permet le choix d'un fragment particulier, vérifié par verif_cont. verif_cont a 6 paramètres :

- **liste des résultats à démontrer,**
- **type de fragment choisi**
- **choix effectué (numéro de fragment),**
- **bloc courant avant le choix,**
- **liste de résultats pouvant être démontrés dans le bloc courant**
- **nouvelle liste de résultats pouvant être démontrés après le choix.**

Vérification d'un choix d'un résultat à prouver :

```
verif_cont(_,4,C,_,[C],[C]):- !.
verif_cont(_,4,C,_,R,[C]):-member(C,R), !.
verif_cont(_,4,C,_,R,_):-prouve(C,[D]),member(D,R), !,mess_err(D,1).
verif_cont(_,4,C,_,[R],_):-prouve(C,L),member(R,L), !,mess_err(R,1).
verif_cont(_,4,C,Bloc,[R],_):-  eval(type_aide,TA),less_than(1,TA),
                                prouve(C,[H|_]),disjoint([H],Bloc),
                               !,mess_err(H,4).
verif_cont(_,4,_,_,_,_):- !,demon(16).
                /* résultat non démontrable avec le reste */

Vérification du choix d'un autre type de fragment~:

verif_cont(_,_,C,Bloc,_,_):-member(C,Bloc), !,demon(17).
               /* fragment donné deux fois  */

verif_cont(_,_,C,Bloc,[R],[R]):-prouve(R,L),inclus(Bloc,[R|L]),member(C,L), !.

/* verif_cont(AD,_,C,[R],[R],_):- C ne demontre pas R */
verif_cont(AD,_,C,Bloc,[R],_):-eval(type_aide,TA),eq(TA,3),
                         mem_gen(Res,AD),prouve(Res,L),member(C,L),
                         neq(C,R), !,mess_err(Res,3).
verif_cont(_,_,C,Bloc,[R],_):-member(R,Bloc), !,demon(18).
                         /* résultat connu, mauvaise démonstration */
verif_cont(_,_,C,Bloc,R,R1):-dans_preuve([C|Bloc],R,R1),neq(R1,[]), !.
verif_cont(_,_,_,_,_,_):-demon(19).
                         /* résultat non connu, prémisses non liées */
```

D.d.4 Vérification connecteur en cours

Le prédicat verif_connec a 6 paramètres :

- choix effectué (le connecteur),
- bloc courant,
- liste de résultats pouvant être démontrés dans le bloc courant
- nouvelle liste de résultats pouvant être démontrés après le choix.
- attente d'un connecteur éventuel,
- nouvelle attente générée sur le type du fragment suivant.

Choix erroné d'un connecteur de fin de bloc :

```
verif_connec('.',_,_,_,_,_,_):-~ !,demon(9).
verif_connec(';',_,_,_,_,_,_):-~ !,demon(9).
verif_connec(ou,_,_,_,_,_,_):-~ !,demon(10).
```

Choix de 'car' :

```
verif_connec(car,_,RC,RC,[car],[1,2,3]):-~ !.
verif_connec(car,[X|_],_,_,_,_,_):- !,enonce(X,T,_),
                        string_from(T,T1),concat("5",T1,T2),
                        string_from(T3,T2),demon(T3).
verif_connec(_,_,_,_,[car],_,_):- !,demon(6).
```

Choix de 'donc' (une condition nécessaire et suffisante est que le résultat courant que l'on cherche à montrer ne soit pas encore cité, ce qui n'est plus le cas en fin de démonstration).

```
verif_connec(donc,Bloc,[R],_,_,_,_):-member(R,Bloc), !,demon(7).
verif_connec(donc,_,R,R,_,[4]):- !.
```

Autre choix :

```
verif_connec(_,Bloc,[R],[R],_,[1,2,3]):-member(R,Bloc), !.
                /* ceci assure que le résultat n'est pas encore cité~ !*/
verif_connec(_,Bloc,R,_,_,_,_):-doit_prouver(R,Bloc), !,demon(8).
verif_connec(_,Bloc,R,R,_,[1,2,3]).
```

D.d.5 Vérification de fin de bloc

La vérification du connecteur en fin de bloc est très particulière. Elle peut consister dans le choix d'un indicateur de fin (le point ou le point-virgule) ou s'intégrer partiellement au bloc suivant, les d'un éléments du bloc courant devant être repris dans le bloc suivant bien que n'étant pas spécifiquement déclaré.

`verif_fin` comprend 7 paramètres :

- liste des résultats à démontrer,
- choix effectué (le connecteur),
- le résultat venant d'être démontré,
- le dernier fragment du bloc courant,

- **le nouveau bloc à construire,**
- **nouvelle liste de résultats pouvant être démontrés après le choix,**
- **attente générée sur le type du fragment suivant.**

```
verif_fin(_,'.',_,_,[],[],nil):-!.
verif_fin(_,';',_,_,[],[],nil):-!.

verif_fin(AD,et,Res,Res,L,[R1],[4]):-   /* X => Y1 et X => Y2 */
            prouve(Res,L),prouve(R1,L),member(R1,AD),!.  /*1 seule preuve*/
verif_fin(_,et,Res,Res,_,_,_):-!,demon(23).

verif_fin(_,car,Res,Res,_,_,_):-eval(type_verif,N),eq(4,N),!,demon(24).
verif_fin(_,car,Res,Res,_,_,_):-!,demon(20).
verif_fin(_,car,_,X,_,_,_):-!,enonce(X,T,_),
                        string_from(T,T1),concat("5",T1,T2),
                        string_from(T3,T2),demon(T3).

verif_fin(AD,ou,Res,Res,[Res],[R1],[4]):-
                prouve(R1,[Res]),member(R1,AD),!.

verif_fin(AD,donc,Res,_,[Res],[R1],[4]):-
                eval(type_verif,N),less_than(2,N),
                mem_gen(R1,AD),prouve(R1,L),
                member(Res,L),disjoint(L,AD),!.

verif_fin(AD,donc,Res,_,[Res],[R1],[4]):-
                prouve(R1,[Res]),member(R1,AD),!.

verif_fin(AD,_,Res,_,_,_,_):-
                prouve(R1,[Res]),member(R1,AD),!,demon(25).

verif_fin(AD,or,Res,DER,[Res],[R1],[1,2,3]):-
                eval(type_verif,N),less_than(1,N),v(N,DER,Res),
                mem_gen(R1,AD),prouve(R1,L),
                member(Res,L),disjoint(L,AD),!.

v(2,DER,Res):-!,neq(DER,Res),fail.
v(_,_,_).

verif_fin(_,Choix,_,_,_,_,_):-
                member(Choix,[ou,donc]),!,demon(22).

verif_fin(_,_,_,_,_,_,_):-eval(type_verif,N),eq(4,N),!,demon(24).
verif_fin(_,_,_,_,_,_,_):-demon(20).
```

Annexe D. Le système de vérification de la démonstration dans ARRIA

Annexe E

Programme de diagnostic

E.a Description de BADAUD-FRACT

Le prédicat `regle_simplif` concerne les règles de simplification d'une fraction connue sous une forme évaluée (ne nécessitant pas de calcul intermédiaire). Syntaxe : `regle_simplif(T,[N,D],[LDN,LDD,GCD,LDGCD],ResPart,E)` où les différents paramètres sont :

- type de simplification effectuée,
- fraction de départ (liste numérateur, dénominateur),
- liste de 4 éléments :
  - liste des diviseurs du numérateur,
  - liste des diviseurs du dénominateur,
  - pgcd des numérateur et dénominateur,
  - liste des diviseurs du pgcd,
- résultat partiel obtenu,
- liste des types d'erreurs effectuées.

```
regle_simplif(_,[N,D],_,[N1,D1],[st(u)]):-   /* ou s euclidienne par 10 */
        N > 10,D > 10,                       /*     72/32 --> 7/3       */
        A is N mod 10,A is D mod 10,
        N1 is N / 10,D1 is D / 10.

regle_simplif(_,[N,D],_,[N1,D1],[st(d)]):-   /*    ou soustraction      */
        N > 10,D > 10,                       /*     74/79 --> 4/9       */
        A is N / 10,A is D / 10,
        N1 is N mod 10,D1 is D mod 10.

regle_simplif(_,[N,D],[_,_,GCD,_],[N1,D1],[se(Div)]):-
        member(Div,[2,4,8,16,32,64]),          /* s euclidienne par 2 */
        0 =\= GCD mod 2,                       /*    21/34 --> 10/17  */
        N1 is N / Div,0 =\= N1,
```

```
        D1 is D / Div,0 =\= D1.

regle_simplif(directe,[N,D],_,[1,S],[ss(N)]):-  /* soustraction entière */
         N < D,S is D - N.                      /*    21/34 --> 1/13    */

regle_simplif(_,[N,D],[LDN,_,GCD,_],[N1,D1],[se(Div,num)]):-
        member(Div,LDN),2 =\= Div,              /* s.euclid. numérateur */
        premiers(Div,GCD),                      /*    21/34 --> 7/11    */
        N1 is N / Div,
        D1 is D / Div.

regle_simplif(_,[N,D],[_,LDD,GCD,_],[N1,D1],[se(Div,den)]):-
        member(Div,LDD),2 =\= Div,              /* s. euclid. denom.  */
        premiers(Div,GCD),                      /*   34/21 --> 11/7   */
        N1 is N / Div,
        D1 is D / Div.

regle_simplif(directe,[N,D],[LDN,LDD,GCD,_],[N1,D1],[sh(DN,DD)]):-
        member(DN,[1|LDN]),                     /*  s. hétérogène    */
        member(DD,[1|LDD]),premiers(DN,DD),     /*  25/14 --> 5/2    */
        N1 is N / DN,
        D1 is D / DD.

regle_simplif(_,[N,D],[_,_,_,LDGCD],[N1,D1],[rqdn(D)]):- /* Remplace quot. */
        member(N1,LDGCD),N1 =\= D,                        /*   par div.     */
        D1 is D / N1.                                     /*  75/45 --> 5/9 */

regle_simplif(_,[N,D],[_,_,_,LDGCD],[N1,D1],[rqdd(N)]):- /* Remplace quot. */
        member(D1,LDGCD),D1 =\= N,                        /*    par div.    */
        N1 is N / D1.                                     /* 75/45 --> 25/3 */
```

Certaines erreurs de calcul sont prises en compte :

- simplifications partielles avant effectuation :

```
simplif_partielle([X,Y],D,N1,D1,E1):-
                pgcd(X,D,X1),X1 =\= 1,
                liste_div(X1,LX1),
                member(Div,LX1),
                X2 is X / Div,
                D2 is D / Div,
                re_eval(X2,Y,D2,N1,D1,E1).

simplif_partielle([X,Y],D,N1,D1,[sae]):-
                pgcd(Y,D,Y1),Y1 =\= 1,
                liste_div(Y1,LY1),
                member(Div,LY1),
                Y2 is Y / Div,
                D1 is D / Div,
                N1 is X + Y2.

re_eval(X2,Y,D2,N1,D2,[sae]):-
        N1 is X2 + Y.
```

Annexe E. Programme de diagnostic

```
re_eval(X,Y,D,N1,D1,[dsae]):-
        pgcd(Y,D,Y1),Y1 =\= 1,
        liste_div(Y1,LY1),
        member(Div,LY1),
        Y2 is Y / Div,
        D1 is D / Div,
        N1 is X + Y2.
```

- **erreurs de calcul dans les sommes et produits :**

  `faire_produit, faire_somme` :

  - **1er nombre,**
  - **2ème nombre,**
  - **résultat,**
  - **liste d'erreurs effectuées.**

```
faire_produit(X1,Y1,N1,[]):-N1 is X1 * Y1.
faire_produit(X1,Y1,N1,[ec(table)]):-voir_table(X1,Y1,N1).
faire_produit(X1,Y1,N1,[ec(m)]):-N2 is X1 * Y1,faire_ec(N1,N2).
faire_produit(X1,Y1,N1,[ec(m)]):-X is X1 - 1,N1 is X * Y1.
faire_produit(X1,Y1,N1,[ec(m)]):-X is X1 + 1,N1 is X * Y1.

faire_somme(X1,Y1,N1,[]):-N1 is X1 + Y1.
faire_somme(X1,Y1,N1,[ec(a)]):-N2 is X1 + Y1,faire_ec(N1,N2).

erreur_calcul(prod,D1,Div,D):-voir_table(D1,Div,D).
erreur_calcul(prod,D1,Div,D):-D2 is D / Div,faire_ec(D1,D2).

voir_table(X,Y,Z):-table(X,Y,Z),!.
voir_table(X,Y,Z):-table(Y,X,Z).

faire_ec(M,Bon):-       ajout(X),M is Bon + X.
faire_ec(M,Bon):-       ajout(X),M is Bon - X,M > 1.

ajout(1).
ajout(2).
ajout(10).

/* Table  partielle des erreurs les plus courantes  (* ou /) : */

table(6,8,46).
table(5,11,75).
table(5,13,45).
table(4,14,66).
```

E.b Exemple du traitement d'un exercice

On considère la série F4 : $(6 * 4 + 18)/72$
(exemple qui nécessite deux pas de calcul au numérateur avant d'effectuer la simplification).

A part les résultats corrects, on trouve 28 types de réponses erronées (hormis les réponses données avec des nombres décimaux)

Irréductibles : 15/13, 5/4, 25/36 (chaque faute ne correspond qu'à un seul élève).
Dénominateur constant (règles non introduites) : 1568/72, 20/72, 62/72 (peut-être erreur de calcul de 20)

Tous les autres résultats conduisent à l'obtention d'un cheminement plausible :

12/7	inv
2/3	sh(7,8), se(7,num), ec (16/24), ec(m) (48/72)
7/2	sh(1,6), rqdd, ec, dsae,
1/2	sh(7,6), rqdn, se(8), st(u), ec(m)
5	dsae
7	sh(1,12) ou dsae
42	sh(1,72)
1/3	sh(7,4), se(4)
11/18	ec ou ec(a) ou ec(m) 44/72,
14/34	ec
15/18	sae (30/36)
23/36	ec
3/12	sh(7,3) ou rqdn
31/36	ec
4/7	st(u)
4/9	ec(a) ou ec(m)
43/72	ec(a)
6/7	rqdd
7/13	ec
7/18	sh(3,2)
7/72	sh(6,1)
7/8	sh(2,3)

Annexe F

Structure de CAMELEON

F.a Les objets

On distingue deux types principaux : les constantes et les fonctions, contenant ou ne contenant pas de variable.
Exemple : (Log 2 + $\sqrt{5}$) est une constante réelle positive.

On distingue :

- Nombres entiers et réels
- Variables utilisées : elles ont pour nom X, Y ou Z
- Paramètres ou variables entières :
  - N quelconque
  - P pair
  - I impair
- Paramètres réels : A1 B1
- Fonctions inconnues : F G

On a différents types d'opérateurs (considérés aussi comme des fonctions) :

- les opérateurs simples : minus, abs, $\sqrt{}$, log, exp
- les opérateurs paramétrés : ^ N (N est entier)
- les fonctions naires : * et +
- les relations : =, >, <

Propriétés à considérer :
- constantes : signe (postif, négatif, nul), sous-type (réel, entier, nil), parité (pair, impair) pour entier, évaluation (valeur ou valeur approchée).

- fonctions : type, definition, image (donne le signe), monotonie, zéros, parité, fonction réciproque.

F.b Exemples de déclaration sur les fonctions

```
/*-------------------------------------------------------------------------
            Ensemble image des fonctions standards
-------------------------------------------------------------------------*/

image([X,Y],['=',X,Y]):-constante(Y).
image([X,X],[]).
image([X,[F,X]],[]):-member(F,[minus,log]).
image([X,[exp,X]],['>',X,0]).
image([X,[F,X]],[non,['<',X,0]]):-member(F,[abs,'√']).
image([X,['^',X,N]],[non,['=',X,0]]):-signe(N,negatif),parite(N,impair).
image([X,['^',X,N]],['>',X,0]):-signe(N,negatif),parite(N,pair).
image([X,['^',X,N]],[non,['<',X,0]]):-signe(N,positif),parite(N,pair).
image([X,['^',X,N]],[]):-signe(N,positif),parite(N,impair).


/*-------------------------------------------------------------------------
                  Parité des fonctions standards
-------------------------------------------------------------------------*/

parite([X,Y],pair):-constante(Y).
parite([X,X],impair).
parite([X,[abs,X]],pair).
parite([X,[minus,X]],impair).
parite([X,['^',X,N]],pair):-parite(N,pair).
parite([X,['^',X,N]],impair):-parite(N,impair).


/*-------------------------------------------------------------------------
                Monotonie des fonctions standards
-------------------------------------------------------------------------*/

monotonie([X,Y],constante):-constante(Y).
monotonie([X,X],croissante).
monotonie([X,[F,X]],croissante):-member(F,[exp,log,'√']).
monotonie([X,[minus,X]],decroissante).
monotonie([X,[abs,X]],[[decroissante,['<',X,0]], [croissante,['>',X,0]]]).
monotonie([X,['^',X,N]],croissante):-signe(N,positif),parite(N,impair).
monotonie([X,['^',X,N]],decroissante):-signe(N,negatif),parite(N,impair).
monotonie([X,['^',X,N]],[[decroissante,['<',X,0]], [croissante,['>',X,0]]]):-
        signe(N,positif),parite(N,pair).
monotonie([X,['^',X,N]],[[croissante,['<',X,0]], [decroissante,['>',X,0]]]):-
        signe(N,negatif),parite(N,pair).
```

F.c Exemples de tableaux de comportement

Les déclarations de propriétés correspondent :
- aux opérateurs considérés comme fonctions,
- aux diverses associations possibles :
  - opérateurs n-aires : somme, produit ;
  - composition d'opérateurs (loi ° sous-entendue dans l'écriture imbriquée).
  - comportement des propriétés et des valeurs des propriétés avec l'application des différents opérateurs.
  - comportement des propriétés et des valeurs des propriétés avec l'application d'opérateurs ou de fonctions ayant des propriétés connues.

Ainsi, par exemple, l'opérateur MINUS appliqué à une fonction impaire donne une fonction paire :
- MINUS (impaire) --> paire
- F(impaire) avec F impaire --> paire

En fait, l'ensemble de ces types de relation permet d'effectuer de très nombreuses déductions.

On arrive à des tableaux génériques de comportement tels que :

```
/*-------------------------------------------------------------------------
                        Règle des signes
-------------------------------------------------------------------------*/

regle_des_signes(_,_,['+',X,X],X).
regle_des_signes(_,_,['+',_,_],nil).

regle_des_signes(P,_,['*',X,X],P).
regle_des_signes(_,N,['*',_,_],N).

regle_des_signes(P,_,['^',P,_],P).
regle_des_signes(P,N,['^',N,C],P):-parite(C,pair).
regle_des_signes(_,N,['^',N,_],N).
```

Annexe F. Structure de CAMELEON

Bibliographie

[ABELSON & diSESSA 80] H. Abelson et A. diSessa
" Turtle Geometry : the Computer as a Medium for Exploring Mathematics "
MIT Press, Cambridge, 1980.

[ABELSON & SUSSMAN 85] Harold Abelson et Gerald Jay Sussman (avec Julie Sussman)
" Structure and Interpretation of Computer Programs "
M. I. T. Press, 1985.

[AIELLO & Al. 88a] L. Aiello, M. Carosio, A. Micarelli
" An ITS for the study of mathematical functions "
in ITS-88, Montréal, p. 170-175, 1988.

[AIELLO & Al. 88b] L. Aiello, M. Carosio, A. Micarelli
« Le projet d'un système intelligent de formation en mathématiques : SEDAF »
Summer University Le Mans 1988.

[AKSCYN & Al. 88] R. M. Akscyn, D. L. McCracken, E. A. Yoder
" KMS : a distributed Hypermedia system for managing knowledge in organizations "
CACM, july 1988.

[AL-KADURIE 88] O. M. Al-Kadurie
" A tutoring and diagnostic system for arithmetic "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 352-355, 1988.

[ALDERMAN & MAHLER 77] D. L. Alderman, W. A. Mahler
" Faculty acceptance of instructional technology : attitude towards educational practices and CAI at community college "
Prog. Learning and Ed. Tech., 14, p. 77-87, 1977.

[ALPERT 75] D. Alpert
" The PLATO IV system in use : a progress report "
in O. Lecarme and R. Lewis (eds.) *" Computers in Education "*, Amsterdam, North Holland, 1975.

[ALSCHULER 89] Liora Alschuler
" Hand-crafted Hypertext-Lessons from the ACM Experiment "
in The Society of Text, Edward Barett ed. MIT Press, p. 343-361, 1989.

[ANDERSON 83a] J. R. Anderson
" The architecture of cognition "
Cambridge Mass. : Harvard University Press, 1983.

[ANDERSON 83b] Anderson J. R.
" Acquisition of proof skills in geometry "
in J. G. Carbonell, R. Michalski & T. Mitchell *" Machine Learning, an A. I. approach "* Springer Verlag, p. 191-219, 1983.

[ANDERSON 87] J. R. Anderson
" Production Systems, Learning and Tutoring "
in Production System Models of Learning and Development (D. Klahr, P. Langley, R. Neches, eds), p. 437-458, 1987.

[ANDERSON 88] J. R. Anderson
" The Expert Module "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 21-54, 1988.

[ANDERSON 89] J. A. Anderson
" Psychology and Intelligent Tutoring "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 1, 1989.

[ANDERSON & Al. 85] Anderson J. R., Boyle C. F. & Yost G.
" The geometry tutor "
Proceedings of the Ninth International Joint Conference on A. I. (p 1-7) Los Altos, CA : Morgan Kaufmann

[ANDERSON & REISER 85] J. R. Anderson, B. J. Reiser
" The LISP Tutor "
Byte, vol. 10, n°4, p. 159-175, 1985.

[ARTIGUE & Al. 85] M. Artigue, V. Gautheron et E. Isambert
« Analyse non standard et enseignement »
Cahier de didactique des Mathématiques n° 15 IREM Paris VII, 1985

[ASPETSBERGER & KUTZLER 88] K. Aspetsberger and B. Kutzler
" Symbolic computation - A new chance for Education "
in Computers in Education Elsevier, IFIP p. 331-336, 1988.

[ATKINSON 76] R. C. Atkinson
" Adaptative instructional systems : some attempts to optimize the learning process "
in D. Klahr (ed.), *" Cognition and Instruction "* Hillsdale, NJ : Erblaum, 1976.

[ATTISHA & YAZDANI 83] M. Attisha, M. Yazdani
" A micro-computer based tutor for teaching arithmetic skills "
Instructional Science, 12, p. 333-342, 1983.

[BACHELARD 67] G. Bachelard
« La formation de l'esprit scientifique »
Vrin, 1967.

[BALACHEFF 78] N. Balacheff
« Les graphes de démonstration : outil pour l'étude des démonstrations naturelles »
Thèse de 3ème cycle, Institut National Polytechnique de Grenoble, 1978

[BALACHEFF 87] N. Balacheff
« Processus de Preuve et Situations de Validation »
Eduactional Studies in Mathematics vol. 18, p. 147-176, 1987.

[BALPE 90] Jean Pierre Balpe
« Hyperdocuments, Hypertextes, Hypermedias »
Eyrolles, 1990.

[BARATAUD & BRUNELLE 85] Lucien Brunelle et Dominique Barataud
« De l'erreur à la réussite en Mathématiques »
Nathan, 1985.

[BARON 82] Monique Baron-Leleu
« Un système pour exprimer et mettre en oeuvre des connaissances en manipulation formelle d'expressions »
Thèse de 3ème cycle, Paris VI, Décembre 82.

[BARR & FEIGENBAUM 81] Avron Barr et Edward A. Feigenbaum
" The handbook of Artificial Intelligence "
Volume I, William Kaufmann, Inc. 1981 (Traduction française 1986, Editions Eyrolles).

[BARR & FEIGENBAUM 82] Avron Barr et Edward A. Feigenbaum
" The handbook of Artificial Intelligence "
Volume II, William Kaufmann, Inc. 1982.

[BARRETT 88] E. Barrett
" Introduction : A New Paradigm for Writing with and for the Computer "
in E. Barrett (ed) *" Text, ConText, and HyperText "*, 1988.

[BARRETT 89] Edward Barrett
" Introduction : Thought and Language in a Virtual Environment "
in The Society of Text, Edward Barett (ed.), MIT Press, 1989.

[BARRETT & PARADIS 88] E. Barrett, J. Paradis
" The On-line Environment and In-House Training "
in E. Barrett (ed) *" Text, ConText, and HyperText "*, p. 227-249, 1988.

[BARRIER 90] Emile Barrier
« Enquête internationale sur l'utilisation des ordinateurs dans l'enseignement. Premiers résultats français »
in Education & Pédagogies, CIEP, mars 90

[BARRIL 87] Patrick Barril
« Explications et comptes rendus d'erreurs en programmation logique »
Université PARIS VI Colloque E. A. O. 87.

[BARRIL 88] P. Barril
« Bases de connaissances pour l'enseignement assisté par ordinateur »
Laboratoire MASI, Université Paris VI, 1988.

[BARRIL 89] P. Barril
« DIPLOMAT : un logiciel intelligent pour l'apprentissage de la programmation logique »
Thèse de l'Université Pierre et Marie Curie, Paris, 1989.

[BARROS 90] L. A. Barros
" Hypertext for learning - design criteria "
in The Seventh International Conference on Technology and Education CEP Consultants LTD, p. 394-97, 1990.

[BARUK 73] S. Baruk
« L'âge du capitaine. De l'erreur en mathématiques »
Seuil, 1973.

[BAULAC 90] Y. Baulac
« Un micromonde de géométrie, Cabri-géomètre »
Thèse d'Université, Grenoble I, février 1990.

[BEAULIEU & Al. 89] H. Beaulieu, A. Baronian, J. C. Bélanger
« Vers un tuteur intelligent enseignant une stratégie pour l'intégration mathématique »
Proceedings of the sixth Canadian Symposium on Instructional Technology p. 54-57, 1989

[BEESON 89] M. J. Beeson
" The User Model in MATHPERT : an Expert System for Learning Mathematics "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 9-14, 1989.

[BENTO & COSTA 88] C. Bento and E. Costa
" Automatic Cognitive Modelling in Hierarchical Domains "
ECAI, 88.

[BERGERON 88] Anne Bergeron
« Environnements d'exploration, représentation et traitement de l'information »
in ITS-88, Montréal, p. 54-60, 1988.

[BERGERON & BORDIER 90] A. Bergeron et J. Bordier
" An Intelligent Discovery Environment for Probability and Statistics "
International Conference on ARCE, Tokyo, p. 167-173, juillet 1990.

[BERGMAN & KANOUI 78] M. Bergman, H. Kanoui
« SYCOPHANTE : système de calcul formel sur ordinateur »
Rapport final ATP informatique. Groupe IA, Marseille, 1978.

[BESSIERE & Al. 89] C. Bessiere, M. Giry, J. L. Leonhardt
" Advanced authoring tools "
Publication DELTA, Multimedia Journal 1, Intermaps, 1989.

[BLEDSOE 71] W. W. Bledsoe
" Splitting and reduction heuristics in automatic theorem proving "
Artificial Intelligence, 2, p. 55-77, 1971.

[BLEDSOE & Al. 85] W. W. Bledsoe, K. Kunen, R. Shotsak
" Completness results for inequality provers "
Artificial Intelligence, p. 255-288, 1985.

[BLEDSOE & HINES 80] W. W. Bledsoe, L. M. Hines
" Variable elimination and chaining in a resolution-based prover for inequalities "
5th Conference on Automated Deduction, Les Arcs, p. 77-87, 1980.

[BLONDEL & Al. 90] F. M. Blondel, M. Schwob, M. Tarizzo
" A problem solving environment for quantitative chemistery "
International Conference on ARCE, Tokyo, p. 111-116, juillet 1990.

[BOBROW 64] D. Bobrow
" A question answering system for high school algebra word problems "
Proceedings of the Fall Joint Conference, p. 591-614, 1964.

[BOBROW & STEFIK 83] Daniel G. Bobrow, Mark Stefik
" The LOOPS Manual "
Xerox Corporation, 1983

[BOCKER & Al. 89] H. D. Böcker, J. Herczeg, M. Herczeg
" ELAB - An Electronics Laboratory "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 15-24, 1989.

[BONAR & CUNNINGHAM 88a] J. Bonar & R. Cunningham
" Bridge : an intelligent tutor for thinking about programming "
in *" A. I. and Human Learning "* (ed. J. Self) Chapman and Hall Computing, p. 391-409, 1988.

[BONAR & CUNNINGHAM 88b] J. G. Bonar and R. Cunningham
" Intelligent tutoring with intermediate representation "
in ITS-88, Montréal, p. 25-32, 1988.

[BONAR & SOLOWAY 85] J. Bonar, E. Soloway
" Pre-programming knowledge : A major source of Misconceptions in Novice Programmers "
Human-Computer Interaction, 1985.

[BONNET & Al. 81] A. Bonnet, M. O. Cordier, D. Kayser
" An ICAI system for teaching derivates in Mathematics "
in R. Lewis and E. D. Tagg (eds), *" Computers in Education "*, Amsterdam, North Holland, 1981.

[BONNET & Al. 85] C. Bonnet, J. M. Hoc et G. Tiberghien
« Psychologie, intelligence artificielle et automatique »
Pierre Mardaga, 1985.

[BORK 88] A. Bork
" Technology in Education : Moving from Trivial to Signifiant Impact "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 5-8, 1988.

[BORNE 83] Isabelle Borne
« Les objets dans Logo »
BIGRE n°37, p. 2-10, décembre 1983.

[BORNE & LEMOYNE 87] I. Borne et G. Lemoyne
« Applications d'un environnement objet à la représentation imagée des connaissances pour la résolution de problèmes arithmétiques »
Colloque E. A. O. 87, Mars 87.

[BOULET & Al. 89a] M. M. Boulet, L. Barbeau, S. Slobodrian
« Conception d'un module d'intervention d'un système conseiller »
Proceedings of the sixth Canadian Symposium on Instructional Technology p 80-84, 1989

[BOULET & Al. 89b] M. M. Boulet, P. Duchastel, J. W. Brahan, R. A. Orchard, A. T. Parent, C. S. Phan, A. J. Cole, J. R. Hatley, R. Pilkington *" A Design Task Advisor "*
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 25-31, 1989.

[BOURGOIN 79] D. Bourgoin
« Organisation des connaissances dans un programme résolvant des exercices d'arithmétique »
Colloque I. A., Rouen, Publication du G. R. 22, p. 113-131, septembre 1979.

[BOUVIER 81] Alain Bouvier
« La mystification mathématique »
Hermann, 1981.

[BROUAYE & Al. 87a] Brouaye, Bruillard, Marchal, Weidenfeld
« SEVE, un système auteur pour le traitement de documents »
Colloque E. A. O. 87, Mars 87.

[BROUAYE & Al. 87b] Brouaye, Bruillard, Ferret, Weidenfeld
« APPAT, un tuteur intelligent pour l'apprentissage des tableurs par la résolution de problèmes »
Colloque E. A. O. 87, Mars 87.

[BROWN 85] J. S. Brown
" Process versus product : a perspective on tools for communal and informal electronic learning "
Journal of Educational Computing Research, 1, p. 179-201, 1985.

[BROWN & BURTON 78] J. S. Brown, R. R. Burton
" Diagnostic Models of Procedural Skills in Basic Mathematical Skills "
Cognitive Science, Vol. 2, p. 155-192, 1978.

[BROWN & VANLEHN 80] J. S. Brown, Kurt VanLehn
" Repair theory, a generative theory of Bugs in Procedural Skills "
Cognitive Science 4, p. 79-426, 1980.

[BROWN & VANLEHN 82] J. S. Brown, Kurt VanLehn
" Towards a generative theory of bugs "
in Carpenter, Moser, Romberg (eds), *" Addition and subtraction : a cognitive perspective "*, LEA, p. 117-135, 1982.

[BROUSSEAU 83] G. Brousseau
« Les obstacles épistémologiques et les problèmes en mathématiques »
Recherche en Didactique des mathématiques, 1983. vol 4.2, p. 172

[BRUILLARD 86a] E. Bruillard
« PROD : système de production pour la transformation de mots »
INRP / DP5, CNDP-ULE, 1986.

[BRUILLARD 86b] E. Bruillard
« PROLOGO : un PROLOG en LOGO »
CNDP-ULE, 1986.

[BRUILLARD 86c] E. Bruillard
« FORMATEXTE : formateur de texte en LOGO »
CNDP-ULE, 1986.

[BRUILLARD 88] E. Bruillard
« Utilisation du système SEVE pour la conception de didacticiels »
Summer University Le Mans 1988.

[BRUILLARD & ROCLAND 86] E. Bruillard et N. Rocland
« Simulation d'interpréteur Logo »
Rapport interne CMI, Paris, 1986.

[BRUILLARD & PEREIRA 87] E. Bruillard et I. Péreira
« Quelques problèmes d'apprentissage avec Logo et Prolog »
Colloque E. A. O. 87, Mars 87.

[BRUILLARD & WEIDENFELD 90] E. Bruillard, G. Weidenfeld
" Some examples of Hypertext's Applications "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[BUNDY 75] Alan Bundy
" Analysing mathematical proofs "
DAI Research Report n°2, University of Edinburgh, 1975.

[BUNDY 79] Alan Bundy
" A treatise on elementary equation solving "
DAI working paper n°51, University of Edinburgh, 1979.

[BUNDY 83] Alan Bundy
" The computer modelling of mathematical reasoning "
Academic Press, 1983.

[BUNDY 84] Alan Bundy
" Meta-level inference and consciousness "
in The Mind and the Machine, philosophical aspects of artificial intelligence Ellis Horwood, p. 156-167, 1984.

[BUNDY & Al. 79]
" MECHO : a program to solve mechanics problems "
DAI Working Paper n°50, University of Edinburgh, 1979.

[BUNDY & Al. 85] A. Bundy, B. Silver, D. Plummer
" An Analytical Comparaison of Some Rule-learning Programs "
Artificial Intelligence, 27, 1985.

[BUNDY & WELHAM 81] Alan Bundy and Bob Welham
" Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation "
Artificial Intelligence, vol 16, n°2, may 1981, p. 189-211

[BURNS & CAPPS 88] H. L. Burns, C. G. Capps
" Foundations of ITS : an Introduction "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 1-20, 1988.

[BURTON 82] Richard R. Burton
" Diagnosing bugs in a simple procedural skill "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 157-183, 1982.

[BURTON 88] R. R. Burton
" The Environment Module of ITS "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 109-142, 1988.

[BURTON & BROWN 82] Richard R. Burton and J. S. Brown
" An investigation of computer coaching for informal learning activities "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 157-183, 1982.

[BUTHION 75] M. Buthion
« Un programme qui résout formellement des problèmes de construction géométrique »
Thèse de 3ème cycle, Paris VI, 1975.

[BUSH 45] V. Bush
" As we may think "
Atlantic Monthly, p. 101-108, july 1945.

[BUTLEN 85] D. Butlen
« Apport de l'ordinateur à l'apprentissage des écritures multiplicatives au CE »
Thèse de 3ème cycle, Paris VII, 1985.

[CABROL & Al. 86] D. Cabrol, C. Cachet, R. Cornelius
" A heuristic problem solver : GEORGE "
Computers and Education, 10, 1, p. 81-87, 1986.

[CABROL & Al. 87] D. Cabrol, C. Cachet, R. Cornelius
" Problem solving partner created using Prolog "
in Lewis and E. Tagg (eds), *" A computer for Each Student "*, North Holland, Amsterdam, p. 159-166, 1987.

[CAILLOT 88] M. Caillot
" Modelling the student's errors in the ELECTRE tutor "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 291-299, 1988.

[CARBONNEL 70] J. R. Carbonnel
" AI in CAI : an artificial approach to CAI "
IEEE Transactions on Man-Machine Systems, vol. 11, n°4, p. 190-202, 1970.

[CARLSON 89] Patricia Ann Carlson
" Hypertext and Intelligent Interfaces for Text Retrieval "
in The Society of Text, Edward Barett (ed.), MIT Press, p. 59-76, 1989.

[CARR & GOLDSTEIN 77] B. Carr and I. Goldstein
" Overlays : a theory for computer aided instruction "
MIT, AI memo 406, 1977.

[CARRIERE & Al. 88] E. Carrière, E. Delozanne et M. Vivet
« Les explications dans le résolveur de problèmes du tuteur AMALIA »
Summer University Le Mans 1988.

[CARRIERE & DELOZANNE 89] E. Carrière et E. Delozanne
« Niveaux de connaissances et phases d'apprentissage dans un tuteur intelligent »
Cahier du LAFORIA n°77, Université Paris VI, 1989.

[CAUBISENS 88] P. Caubisens
« MUSILOG : LOGO pour faire de la musique »
INRP, 1988.

[CAUZINILLE-MARMECHE & MATHIEU 88] E. Cauzinille-Marmèche, J. Mathieu
" Experimental Data for the Design of a Microworld-Based System for Algebra "

in H. Mandl, A. Lesgold (eds), " *Learning Issues for ITS* " Springer-Verlag, p. 278-286, 1988.

[CAVINESS 70] B. F. Caviness
" *On Canonical Forms and Simplification* "
Journal of ACM, vol. 17, n° 2, p. 385-396, 1970.

[CAWSEY 86] Alison Cawsey
" *Decimal Addition Bugs : Model, Applications ans Explanations* "
MSc Thesis, University of Edinburgh, 1986

[CHAN & BASKIN 88] Tak-Wai Chan, Arthur B. Baskin
" *'Studying with the prince : the computer as a learning companion* "
in ITS-88, Montréal, p. 194-200, 1988.

[CHARNAY 86] Roland Charnay
« *L'erreur dans l'enseignement des Mathématiques* »
INRP, Rencontres pédagogiques n° 12, p. 9-32, 1986.

[CHASTENET & HOCQUENGHEM 81] J. Chastenet de Géry et S. Hocquenghem
" *Collective use of a micro-computer with graphics to illustrate Mathematics lesson* "
in R. Lewis and E. D. Tagg (eds), " *Computers in Education* ", Amsterdam, North Holland, 1981.

[CHAUDRON 87] L. Chaudron
« *Divers aspects de la démonstration mathématique* »
in Actes de l'Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 15-36, juillet 1987.

[CHI & Al. 89] T. H. Chi, M. Bassok, M. W. Lewis, P. Reimann, R. Glaser
" *Self-Explanations : How Students Study ans Use Examples in Learning to Solve Problems* "
Cognitive Science, 13, p. 145-182, 1989.

[CHOU 88] S. C. Chou
" *Mechanical geometry theorem proving* "
Reidel, Dordrecht, 1988.

[CHOURAQUI & INGHILTERRA 87] Eugène Chouraqui & Carlo Inghilterra
« *Conception d'une base de connaissance orientée objet pour l'E. A. O. de la géométrie* »
Actes du congrès E. A. O. 87 au Cap d'Agde (p. 477-486)

[CLAES 88] Gérard Claes
« *Contribution à l'application de l'I. A. pour l'E. A. O.* »
Thèse de doctorat, Université Paris-sud Orsay, octobre 1988.

[CLAES 89]
« *De la création à l'utilisation de produits éducatifs conçus à l'aide de nouvelles technologies* »
Publication DELTA, Multimedia Journal 1, Intermaps, 1989.

[CLAES et Al. 88] G. Claes, O. Ounis, Z. Razoarivelo, P. Salembier, M. S. Sridharan
« *Starguide, un générateur de système d'autoformation à l'usage de logiciels* »
Revue T. S. I., Vol. 7, n° 1, février 88.

[CLANCEY 86] W. J. Clancey
" The Science and Engineering of Qualitative Models "
KSL, Report 86-27, Standford University, 1986.

[CLANCEY 87] W. J. Clancey
" The role of qualitative models in instruction "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 49-68, 1988.

[CLANCEY & JOERGER] W. J. Clancey and K. Joerger
" A practical authoring shell for apprenticeship learning "
in ITS-88, Montréal, p. 67-73, 1988.

[COLLINS & BROWN 86] A. Collins and J. S. Brown
" Cognitive Apprenticeship : teaching students the craft of reading, writing and mathematics "
in L. B. Resnick (Ed.) *" Cognition and Instruction : Issues and agendas "* Hillsdale, LEA, 1986.

[COLLINS & BROWN 86] A. Collins and J. S. Brown
" The Computer as a Tool for Learning Through Reflection "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 1-18, 1988.

[CONKLIN 87] J. Conklin
" Hypertext : an introduction and survey "
IEEE, septembre 1987.

[COSTA & Al. 87] E. Costa, S. Duchénoy, Y. Kodratoff
" A resolution based method for discovering student's misconceptions "
in *" Intelligent Computer-Aided Instruction "*, J. Self (ed.), Chapman and Hall Computing, p. 156-164, 1987.

[COURTOIS 88] Joël Courtois
« Système intelligent pour l'apprentissage du diagnostic technique »
in ITS-88, Montréal, p. 327-333, 1988.

[COX & Al. 88] B. M. Cox, J. Jenkins and E. Pollitzer
" Understanding and concept acquisition in adaptative I.T.S. "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 78-81, 1988.

[CRAHAY 87] Marcel Crahay
« LOGO, un environnement propice à la pensée procédurale »
Revue française de pédagogie, n° 80, p. 37-56, 1987.

[CROWDER 59] N. A. Crowder
" Automatic tutoring by means of intrinsic programming "
in E. Galanter (ed.), *" Automatic Teaching : the state of art "*, New York, Wiley, 1959.

[CUMMING 90] Geoff Cumming
" Using Artificial Intelligence to achieve natural learning "
International Conference on ARCE, Tokyo, p. 291-302, juillet 1990.

[CUMMING & SELF 89] G. Cumming, J. Self
" Collaborative Intelligent Educational Systems "

in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 73-79, 1989.

[CUPPENS 87] Roger Cuppens
" AM de Douglas B. Lenat : I am that I am "
in Actes de l'Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 155-168, juillet 1987.

[CUPPENS 88] Roger Cuppens
« Faut-il enseigner la logique? »
in Actes de la IIème Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 135-143, juillet 1988.

[CUPPENS 89] Roger Cuppens
« Apports possibles de l'Intelligence Artificielle à la formation des maîtres en mathématiques »
Bulletin de liaison n°3, IREM de Toulouse, février 1989.

[CUPPENS 90] Roger Cuppens
« IA et enseignement de la géométrie »
IREM, 1990

[DANLOS & Al. 85] L. Danlos, S. Guez, S. Sabbagh
« Systèmes d'aide : vous avez dit intelligent? »
Cognitiva, Paris, p. 101-106, juin 1985.

[DAVENPORT & Al. 87] J. Davenport, Y. Siret et E. Tournier
« Calcul formel : Systèmes et algorithmes de manipulation algébrique »
Masson, 1987.

[DAVIS 84] Robert B. Davis
" Learning mathematics, The Cognitive Science Approach To Mathematics Education "
Croom Helm Ltd, 1984.

[DAVIS & LENAT 82] R. Davis, D. B. Lenat
" Knowledge-based systems in Artificial Intelligence "
McGraw-Hill, New-York, 82.

[DAVIS 90] K. Davis
" Student-written hypertext in a university business-writing course "
in The Seventh International Conference on Technology and Education CEP Consultants LTD, p. 132-133, 1990.

[DE CORTE & Al. 87] E. de Corte, L. Verschaffel, H. Schrooten
" Computer simulation as a tool in studying teacher's cognitive activities during error diagnosis in arithmetic "
in 2d International Conference, *" Children in the Information Age "*, Sofia, Bulgarie, 1987.

[DEDE 88] C. Dede
" The role of Hypertext in transforming information into knowledge "
Proceedings of the National Education Computing Conference, p. 99-100, 1988.

[DELFORGE 89] Bernard Delforge
« Un environnement logiciel intelligent pour l'apprentissage de l'informatique »
Thèse de doctorat, Université Pierre et Marie Curie, Paris VI, février 1989.

[DERAMECOURT & Al. 88] G. Deramecourt, J. Douaire, J. C. Guillaume, R. Neyret, J. L. Porcheron
« Micro-ordinateur et Enseignement des Mathématiques : approche de la division »
INRP, Collection rapports de recherches, n°5, 1988.

[DERRY & Al. 88] S. J. Derry, L. W. Hawkes, U. Ziegler
" A plan-based opportunistic architecture for intelligent tutoring "
in ITS-88, Montréal, p. 116-123, 1988.

[DERRY & Al. 89a] S. H. Derry, L. W. Hawkes, H. Kegelmann, D. Holmes
" Fuzzy Remedies to Problems in Diagnostic Modeling "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 81-85, 1989.

[DERRY & Al. 89b] S. H. Derry, L. W. Hawkes, U. Ziegler, T. Diefenbach
" Characterizing the Problem-Solver : A System for On-line Error Detection "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 86-91, 1989.

[DILLENBOURG 88] P. Dillenbourg
" A pragmatic approach of student modelling : Principles and Architecture of PROTO-TEG "
Summer University Le Mans 1988.

[DILLENBOURG 89] Pierre Dillenbourg
« Perspectives Européennes de Recherche en I. A. appliquée à l'Education et à la Formation »
Conférence invitée aux journées nationales *« IA, Perspectives pour l'Education et la Formation »*, Lyon, avril 1989.

[DILLENBOURG & GOODYEAR 89] P. Dillenbourg, P. Goodyear
" Towards reflective tutoring systems : self-representation and self-improvement "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 92-99, 1989.

[DILLENBOURG & Al. 90] P. Dillenbourg, M. Hilario, P. Mendelsohn, D. Schneider
" The Geneva Manifesto of Intelligent Learning Environments "
TECFA Document 90-2, 1990.

[DILLENBOURG & Al. 90] P. Dillenbourg, M. Hilario, P. Mendelsohn, D. Schneider
" Training transfer : a bridge between the theory-oriented and product-oriented approaches to ITS design "
Applica 90, Lille, 1990.

[diSESSA 82] A. A. diSessa
" Unlearning aristotelian physics : a study of knowledge-based learning "
Cognitive Science, vol. 6, p. 37-75, 1982.

[diSESSA 85] A. A. diSessa
" A Principled Design for an Integrated Computational Environment "
in Human-Computer Interaction, LEA, Vol. 1, p. 1-47, 1985.

[diSESSA 87] A. A. diSessa
" Artificial Worlds and Real Experience "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "*, Ablex, p. 55-77, 1987.

[DOLAND 89] Virginia M. Doland
" Hypermedia as an interpretive act "
Hypermedia, Taylor Graham vol. 1, 1, 1989.

[DONADI & Al. 89] M. H. Donadi, C. E. Hughes, J. M. Moshell
" Seperation of powers : Object-oriented user interfaces promise evolutionary rather than revolutionary upgrades "
BYTE, p. 255-262, march 1989.

[DRESCHER 87] G. L. Drescher
" Object-oriented Logo "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 154-165, 1987.

[Du BOULAY 87] Benedict du Boulay
" Intelligent systems for teaching programming "
in *" Artificial intelligents tools in Education "*, P. Ercoli & R. Lewis ed., North Holland, p. 5-15, 1987.

[Du BOULAY & SLOMAN 88] Benedict du Boulay and Aaron Sloman
" Bread today, Jam tomorrow : the impact of AI on Education "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 82-85, 1988.

[Du BOULAY & SOTHCOTT 87] Benedict du Boulay and Christopher Sothcott
" Computers teaching programming : an introductory survey of the field "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "*, Ablex, p. 345-372, 1987.

[DUMONT 84] Bernard Dumont
« Enquête sur les fractions »
Rapport de recherche, Université Paris VII, 1984.

[DUMONT 89a] Bernard Dumont
« Tuteurs Intelligents vs Systèmes pédagogiques à base de connaissances erronées : comment utiliser les erreurs produites par un apprenant »
Proceedings of the sixth Canadian Symposium on Instructional Technology p 143-147, 1989

[DUMONT 89b] Bernard Dumont
« Questionnements et interprétation des erreurs en mathématiques (élaboration de modèles pour la compréhension des comportements de réponse et la construction d'outils pédagogiques à supports technologiques) »
Thèse de doctorat d'Etat, Paris 7, 1989.

[DUMONT 90] Bernard Dumont
" FRACT : a student's error K. B. S. for diagnosis in fraction calculus "
International Conference on ARCE, Tokyo, p. 7-30, juillet 1990.

[DUVAL & EGRET 89] R. Duval, M. A. Egret
« L'organisation déductive du discours : interaction entre structure profonde et structure de surface dans l'accès à la démonstration »
Annales de Didactique et de Sciences Cognitives, 2, p. 25-40, 1989.

[EGRET & DUVAL 89] M. A. Egret, R. Duval
« Comment une classe de 4ème a pris conscience de ce qu'est une démarche de démonstration »
Annales de Didactique et de Sciences Cognitives, 2, p. 41-64, 1989.

[EPSTEIN 87] S. L. Epstein
" Learning and Discovery : One's System Search for Mathematical Knowledge "
Technical Report 87-1, Dept. of Computer Science, Hunter College, 1987.

[EISENSTADT 89] Marc Eisenstadt
" (AI in (Education in AI) : What does 'domain visualisation' have to offer ITS? "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 100, 1989.

[ENGELBART & ENGLISH 68] D. C. Engelbart, W. K. English
" A research center for augmenting human intellect "
AFIPS Conference Proceedings, 33, 1, 1968.

[ENNALS 84] Richard Ennals
" Teaching logic as a computer language in schools "
in Yazdani (ed.) *" New Horizons in Educational Computing "*, Ellis Horwood, p. 164-177, 1984.

[ESCARBAJAL 84] M. C. Escarbajal
« Compréhension et résolution de problèmes additifs »
Psychologie française, 29, p. 247-252, 1984.

[EVANS 86] Nich Evans
" The Future of The Micocomputer in Schools "
MacMillan Education LTD, 1985.

[EVERTSZ 89] Rick Evertsz
" Refinig the student's procedural knowledge through abstract interpretations. "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 101-106, 1989.

[EVERTSZ & ELSOM-COOK 90] Rick Evertsz and Mark Elsom-Cook
" Generating Critical Problems in Student Modelling "
in Elsom-Cook (ed.) *" Guided Discovery Learning "* Paul Chapman Publishing, 1990.

[FABREGA 88] A. Fâbrega
" EUCLIDES I : a program for geometry in CAI "
in Computers in Education Elsevier, IFIP p. 203-206, 1988.

[FERRET & JIMENEZ 87] E. Ferret, C. Jimenez
« A la découverte des méthodes algébriques »
Groupes Didacticiels, Centre Mondial de l'Informatique Colloque E. A. O. 87.

[FEURZEIG 69] W. Feurzeig
" Programming languages as a conceptual framework for teaching mathematics "
BBN, Report n° 1889, 1969.

[FEURZEIG & Al. 81] W. Feurzeig, P. Horwitz, R. S. Nakebar
" Microcomputers in Education "
Report n° 4798, BBN, Cambridge, 1981.

[FEURZEIG 87] W. Feurzeig
" Algebra Slaves and Agents in a Logo-based mathematics curriculum "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 27-54, 1987.

[FISCHER 88] G. Fischer
" Enhancing Incremental Learning Processes With Knowledge-Based Systems "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 138-163, 1988.

[FISCHER & Al. 78] G. Fischer, R. R. Burton, J. S. Brown
" Analysis of skiing as a success model of instruction : Manipulating the learning environment to enhance skill acquisition "
in Proceed. of the 2d national conf. of the Canadian Society for Computational Studies of Intelligence, 1978.

[FISCHER & Al. 84] G. Fischer, A. Lemke, T. Schwab
" Active Help Systems "
in *" Readings on cognitive ergonomics, mind and computers "*, Berlin, Springer Verlag, p. 116-131, 1984.

[FLAVELL 79] J. H. Flavell
" Metacognition and cognitive monitoring "
American Psychologist, 34, p. 906-911; 1979.

[FLORIS & BEVACQUA 89]
" Development and classroom experimentation of interactive geometry exercises "
Journal of Computer Assisted Learning, vol. 5, n°3, september 1989.

[FONTAINE 87] Marie-Danielle Fontaine
« Logiciel d'aide à la résolution de problèmes de géométrie »
Actes de la IVème Ecole d'été de didactique des mathématiques et de l'informatique 30-5 juillet, Orléans, IREM Paris 7, p. 231-233, 1987.

[FOSS 88] C. L. Foss
" Effective browsing in hypertext systems "
Proceedings of RIAO 88 *" User-oriented content-based text and image handling "*, Cambridge, MA :MIT, 1988.

[FRASSON & de la PASSARDIERE 89] C. frasson, B. de la Passardière
« Modélisation de l'étudiant : une approche conceptuelle »
Proceedings of the sixth Canadian Symposium on Instructional Technology p 39-43, 1989.

[GAUTHIER & IMBEAU] G. Gauthier et G. Imbeau
« Un système tutoriel intelligent : modèle théorique et réalisation »
Proceedings of the sixth Canadian Symposium on Instructional Technology p 109-113, 1989

[GELERTNER 63] H. Gelertner
" Realization of a geometry-theorem proving machine, computer and thoughts "
McGraw Hill, p. 135-152, 1963.

[GENESERETH 82] M. R. Genesereth
" The role of plan in intelligent teaching systems "

in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 137-155, 1982.

[GEORGE 87] C. George
« Les théories actuelles de l'apprentissage »
Colloque de la Société Française de Psychologie : les apprentissages - perspectives actuelles, Saint-Denis, janvier 1987.

[GILLET 79] M. Gillet
« Un programme de démonstration automatique en théorie des groupes utilisant beaucoup de connaissances »
Colloque I. A., Rouen, Publication du G. R. 22, p. 133-142, septembre 1979.

[GILMORE 70] P. C. Gilmore
" An examination of the geometry theorem proving machine "
Artificial Intelligence 1, p. 171-187, 1970.

[GILMORE & SELF 88] D. Gilmore and J. Self
" The application of Machine Learning to student modelling "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 179-196, 1988.

[GLIDEN 90] P. L. Gliden
" Towards developing a model of mathematics learning : modeling knowledge restructuring in learning school algebra "
International Conference on ARCE, Tokyo, p. 237-242, juillet 1990.

[GOLDBERG 79] A. Goldberg
" Educational uses of a Dynabook "
Computers and Education, 3, p. 247-266, 1979.

[GOLDSTEIN 82] I. P. Goldstein
" The genetic graph : a representation for the evolution of procedural knowledge "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 51-77, 1982.

[GOLDSTEIN & PAPERT 77] I. P. Goldstein, S. Papert
" Artificial Intelligence, language and the study of knowledge "
Cognitive Science, vol. 1, 1, 1977.

[GOULD & INZER 81]
" A study of TRIP : a computer system for animating time-rate distance problems "
in R. Lewis and E. D. Tagg (eds), *" Computers in Education "*, Amsterdam, North Holland, 1981.

[GRANDBASTIEN 74] M. Grandbastien
« Un programme qui résout formellement des équations trigonométriques par des procédés heuristiques »
Thèse de 3ème cycle, Paris VI, 1974.

[GRANDBASTIEN 88] M. Grandbastien
« Une approche à bases de connaissances pour l'enseignement de la programmation. Conception et réalisation de SAIDA : Système d'Aide à l'Implantation de Données Abstraites. »
Thèse d'état, Université de Nancy I, 1988.

[GREENO 87] J. G. Greeno
" Instructional Representations Based on Research about Understanding "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 61-88, 1987.

[GREENO 90] J. G. Greeno
" Productive Learning Environments "
International Conference on ARCE, Tokyo, p. 1-11, juillet 1990.

[GREVE 89a] S. H. Greve
" A real-time coaching environment for triangle congruence proofs "
in E. Maurer (ed.) *" Computer-Assisted Learning "*, Springer Verlag, p. 150-157, 1989.

[GREVE 89b] S. H. Greve
" Planning proofs : a problem solving coach for geometry "
Proceedings of the sixth Canadian Symposium on Instructional Technology p 164-167, 1989.

[GROEN 84] Guy Groen
" Theories of LOGO "
Pre-proceedings of the 1984 National Logo Conference, p. 49-54, 1984.

[GRUMBACH & NGUYEN-XUAN 87] A. Grumbach et A. Nguyen-Xuan
« Formaliser l'apprentissage humain : modèles connexionistes et symboliques »
Colloque de la Société Française de Psychologie : les apprentissages - perspectives actuelles, Saint-Denis, janvier 1987.

[GUILBAUD 85] G. TH. Guilbaud
« Leçons d'à peu près »
Christian Bourgeois Editeur, 1985.

[GUIN 89] D. Guin
« Réflexions sur les logiciels d'aide à la démonstration en géométrie »
Annales de Didactique et de Sciences Cognitives, 2, p. 89-110, 1989.

[GUIN 90] D. Guin
« Modélisation des connaissances pour un système d'aide à la démonstration géométrique »
Applica 90, Lille, 1990.

[GUIN & ROUSSELOT 87] Dominique Guin & François Rousselot
« Aide à la recherche d'une démonstration »
Actes du congrès E. A. O. 87 au Cap d'Agde (p. 429-438)

[HALASZ 88] F. G. Halasz
" Reflections on Notecards : seven issues for the next generation of Hypermedia systems "
CACM, vol. 31, 7, july 1988.

[HALL 89] Roger Hall
" Qualitative Diagrams : Supporting the construction of algebraic representations in applied problem solving "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 116-122, 1989.

[HAMBURGER & LODGHER 89] H. Hamburger, A. Lodgher
" Exploration in Coinland "
in E. Maurer (ed.) *" Computer-Assisted Learning "* Springer Verlag, p. 158-166, 1989.

[HAMLIN & STEMP 90] M. D. Berlin Hamlin, G. M. Stemp
" The Missing Link : The Use of Scaling Algorithms to Design Hypertext Instructional Materials "
in The Seventh International Conference on Technology and Education CEP Consultants LTD, p. 294-296, 1990.

[HARTLEY & SLEEMAN 73] J. R. Hartley, D. H. Sleeman
" Towards more intelligent teaching systems "
International Journal of Man-Machine Studies, 2, p. 215-236, 1973.

[HEARN 71] A. C. Hearn
" REDUCE 2 : a system and language for algebraic manipulation "
Proceedings SS-SAM, Los Angeles, p. 128-133, 1971.

[HENNESSY & Al. 89] S. Hennessy, T. O'Shea, R. Evertsz, A. Floyd
" An intelligent tutoring system approach to teaching primary mathematics "
Educational Studies in Mathematics 20, p. 273-292, 1989.

[HENNESSY 90] Sara Hennessy
" Why bugs Are Not Enough "
in Elsom-Cook (ed.) *" Guided Discovery Learning "* Paul Chapman Publishing, 1990.

[HERRSTROM & MASSEY 89] D. S. Herrstrom, D. G. Massey
" Hypertext in Context "
in The Society of Text, Edward Barrett (ed.), MIT Press, p. 45-58, 1989.

[HOC 87] Jean Michel Hoc
« Psychologie cognitive de la planification »
Presses Universitaires de Grenoble, 1987.

[HOFSTADTER 79] Douglas Hofstadter
« Gödel, Escher, Bach : les brins d'une guirlande éternelle »
InterEditions, 1985 (traduction française).

[HOFSTADTER 81] Douglas Hofstadter
« Vues de l'esprit »
InterEditions, 1987 (traduction française).

[HOLLAND 88] G. Holland
« Un logiciel de résolution de problèmes de preuve en géométrie utilisé en tant qu'expert d'un système tutoriel »
Actes du 1er colloque franco-allemand de didactique, p. 275-282, 1988.

[HOWE 87] Jim Howe
« Les technologies de l'information : L'arithmétique et les concepts mathématiques »
Rapport O. C. D. E. (CERI), 1987.

[HOWE & Al. 80] J. A. M. Howe, T. O'Shea, F. Plane
" Teaching Mathematics through Logo programming : an evaluation study "
in R. Lewis and E. D. Tagg (eds), *" CAI : Scope, Progress, Limits "*, Amsterdam, North Holland, 1980.

[HOWE & Al. 84] J. A. M. Howe, P. M. Rose, K. R. Johnson, R. Inglis
" Model building, mathematics and Logo "
in Yazdani (ed.) *" New Horizons in Educational Computing "*, Ellis Horwood, p. 54-71, 1984.

[HOYLES 85] Celia Hoyles
" Developing a context for Logo in school mathematics "
LOGO 85, Theoretical Papers, MIT, Cambridge, July 1985.

[HOYLES & Al. 89] C. Hoyles, R. Noss, R. Sutherland
" Designing a LOGO-based microworld for ratio and proportion "
Journal of Computer Assisted Learning, 5, p. 208-222, 1989.

[HOYLES & NOSS 87] C. Hoyles and R. Noss
" Expanding Children's Mathematics with the Computer "
in D. C. Johnson and F. Lovis (eds) *" Informatics and the teaching of mathematics "* North Holland, IFIP, 1987.

[HUTCHINS & NORMAN 85] E. L. Hutchins and D. A. Norman
" Direct manipulation interfaces "
in *" User Centered System Design : New perspectives on Human-Computer Interaction "*, 1985.

[IMBEAU & Al. 88] G. Imbeau, C. Frasson, G. Gauthier
" A Multi-Expert System for Large Scale Intelligent Tutoring "
in ITS-88, Montréal, p. 272-277, 1988.

[IREM 79] Groupe «premier cycle» de l'IREM de Paris 7
« Raisonner »
Bulletin de liaison des Professeurs de Mathématiques IREM Paris 7 n° 23 décembre 79

[IRISH & TRIGG 89] P. M. Irish, R. H. Trigg
" Supporting Collaboration in Hypermedia : Issues and Experiences "
in The Society of Text, Edward Barett (ed.), MIT Press, p. 90-106, 1989.

[JOAB 88] Michelle Joab
« Le système NAIADE : modélisation des connaissances et de leur utilisation dans la résolution de problèmes »
in ITS-88, Montréal, p. 409-414, 1988.

[JOAB & Al. 89] E. Cauzinille-Marmèche, M. Joab, J. Mathieu
" NAIADE, a Knowledge Based System for Explanation "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 42-46, 1989.

[JOAB & Al. 89] M. Joab, E. Cauzinille-Marmèche, J. Mathieu
« NAIADE, un système expert pour dialoguer avec l'élève »
Proceedings of the sixth Canadian Symposium on Instructional Technology p. 58-62, 1989.

[JOHNSON 86] W. L. Johnson
" Intention-based diagnosis of errors in novice programs "
Palo Alto, CA :Morgan Kaufmann, 1986.

[JOHNSON 88] W. L. Johnson
" Modelling programmers' intentions "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 374-390, 1988.

[JOHNSON 88] W. B. Johnson
" Pragmatic Considerations in Research, Development, and Implementation of ITS "

in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 191-208, 1988.

[JOHNSON-LAIRD 83] P. N. Johnson-Laird
" Mental Models "
Cambridge University Press, 1983.

[JONASSEN & GRABINGER 90] David H. Jonassen, R. Scott Grabinger
" Problems and Issues in Designing Hypertext / Hypermedia for learning "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[JONES & Al. 88] J. Jones, M. Millington, P. Ross
" Understanding user behaviour in command-driven systems "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 226-235, 1988.

[JONES & THORNE 88] A. Jones and M. Thorne
" Intelligent Computer aided mathematical problem solving in a modified logo environment "
in Computers in Education, Elsevier, IFIP p. 207-211, 1988.

[JURKOVIC 87] N. Jurkovic
" An intelligent tutor for high-school algebra "
Proceedings of the Fifteenth Annual Computer Science Conference, St Louis, p. 27-31, 1987.

[KAHN 84] Kenneth M. Kahn
" A grammar kit in Prolog "
in Yazdani (ed.) *" New Horizons in Educational Computing "*, Ellis Horwood, p. 178-189, 1984.

[KALTENBACH & FRASSON 89] M. Kaltenbach, C. Frasson
" DYNABOARD : User animated display of deductive proofs in mathematics "
International Journal of Man-Machines Studies 30, p. 149-170, 1989.

[KAMSTEEG & BIERMAN 88] P. A. Kamsteeg and J. Bierman
" An object-oriented prolog and its use in a simulated laboratory for physics "
Summer University Le Mans 1988.

[KAPUT 86] L. Kaput
" Towards a theory of symbol use in mathematics "
in C. Janvier (ed.) *" Problems of representations in the teaching and learning of mathematics "*, Hillsdale, NJ :LEA, 1985.

[KAPUT & CLEMENT 79] . J. Kaput and J. Clement
" Letter to the Editor "
Journal of Children's Mathematical Behavior, vol. 2, n°2, p. 208, 1977.

[KATZ & ANZAI 90] I. R. Katz, Y. Anzai
" The Construction and Use of Diagrams for Problem Solving "
International Conference on ARCE, Tokyo, p. 191-200, juillet 1990.

[KAY & GOLDBERG 77] A. Kay, A. Goldberg
" Personal Dynamic Media "
IEEE Computer, 10, 3, march 1977.

[KELLER 87] R. M. Keller
" Concept Learning in Context "
Proc. of the 4th Workshop on Machine Learning, Irvine, p. 91-102, 1987.

[KIMBALL 82] R. Kimball
" A self-improving tutor for symbolic integration "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 283-307, 1982.

[KODRATOFF 86] Y. Kodratoff
« Leçons d'apprentissage symbolique automatique »
Cepadues-Editions, 1986.

[KODRATOFF & GANASCIA 84] Y. Kodratoff et J. G. Ganascia
" Learning as a non-deterministic but exact logical process "
in The Mind and the Machine, philosophical aspects of artificial intelligence Ellis Horwood, p. 182-191, 1984.

[KOEGEL & Al. 89a] J. Koegel, N. Lakshmipathy, J. D. Schlesinger
" A Tutoring System with Incomplete Domain Expertise "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 123-128, 1989.

[KOEGEL & Al. 89b] J. Koegel, N. Lakshmipathy, J. D. Schlesinger
" An Intermediate Representation for Mathematical Problem Solving "
in E. Maurer (ed.) *" Computer-Assisted Learning "* Springer Verlag, p. 267-281, 1989.

[KONDO & Al. 90] H. Kondo, K. Watanabe, A. Takeuchi, S. Otsuki
" ITS : recognizing the context and guiding processes for fraction calculus "
International Conference on ARCE, Tokyo, p. 31-37, juillet 1990.

[LACOMBE 84] D. Lacombe
« Les composantes du "raisonnement" mathématique »
Colloque de l'ARC, Orsay, 1984, Bulletin de liaison n°2, IREM de Toulouse, p. 87-96, juin 1988.

[LACOMBE 88a] D. Lacombe
« La démonstration formelle. Qu'est-ce-que c'est? A quoi ça sert? »
in Actes de la IIème Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 5-26, juillet 1988.

[LACOMBE 88b] D. Lacombe
« Le pseudo-formalisme scolaire ou l'initelligence artificielle »
in Actes de la IIème Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 27-44, juillet 1988.

[LAKATOS 84] Imre Lakatos
« Preuves et réfutations : essai sur la logique de la découverte mathématique »
Hermann, 1984.

[LANDAU & LESH 83] Richard Lesh et Marsha Landau
" Acquisition of Mathematics Concepts and Processes "
Academic Press, Inc. 1983.

[LANDOW 89a] G. P. Landow
" Course Assignments Using Hypertext : The example of Intermedia "
Journal of Research on Computing in Education, p. 350-365, Spring 1989.

[LANDOW 89b] G. P. Landow
" Hypertext in Literary Education, Criticism, and Scholarship "
Computers and the Humanities, 23, p. 173-198, 1989.

[LANDOW 89c] G. P. Landow
" The rhetoric of Hypermedia : some rules for authors "
Journal of Computing in Higher Education, 1, p. 39-64, 1989.

[LANDOW 90] G. P. Landow
" Popular Fallacies About Hypertext "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[LANGLEY & OHLSSON 84] P. Langley, S. Ohlsson
" Automatic Cognitive Modeling "
in Proceedings of AAAI, Los Altos, CA; Morgan Kaufmann, p. 193-197, 1984.

[LANZ & Al. 83] B. S. Lanz, W. S. Bregar, A. M. Farley
" An intelligent CAI system for teaching equation solving "
Journal of Computer-Based Education, 3, n° 1, p. 35-42, 1983.

[LAUBLET 87] P. Laublet
« Raisonnement mathématique et apprentissage ou Visite guidée d'une intersection »
Publication n° 63, LAFORIA, p. 349-387, septembre 1987.

[LAUBLET 88] P. Laublet
« Apprentissage automatique en mathématique »
in Actes de la IIème Université d'Eté I. A. et enseignement des Maths IREM de Toulouse, p. 111-134, juillet 1988.

[LAURENT 72] J. P. Laurent
« Un programme qui calcule des limites en levant les indéterminations par des procédés heuristiques »
Thèse de 3ème cycle, Paris VI, 1972.

[LAURENT-GENGOUX & Al. 88] P. Laurent-gengoux, H. Lehning, D. Trystram
" On how to conceive software for teaching mathematics "
in Computers in Education Elsevier, IFIP p. 611-614, 1988.

[LAURIERE 76] J. L. Laurière
« Un langage et un programme pour énoncer et résoudre des problèmes combinatoires »
Thèse d'Etat, Paris VI, 1976.

[LAURIERE 87] Jean-Louis Laurière
« INTELLIGENCE ARTIFICIELLE, résolution de problèmes par l'Homme et la machine »
Editions Eyrolles, 1987.

[LAURIERE 88] Jean-Louis Laurière
« INTELLIGENCE ARTIFICIELLE, représentation des connaissances »
Editions Eyrolles, 1988.

[LAURILLARD 90] Diana Laurillard
" Generative Student Models : The Limits of Diagnosis and Remediation "
in Elsom-Cook (ed.) *" Guided Discovery Learning "* Paul Chapman Publishing, 1990.

[LAVOIE & Al. 89] M. Lavoie, M. Gagné, A. Jacques
« Spécifications d'un système informatique servant à la conception et à la réalisation de

logiciels éducatifs à base de connaissances »
Proceedings of the sixth Canadian Symposium on Instructional Technology p 148-154, 1989

[LAWLER 81] R. W. Lawler
" The progressive construction of mind "
Cognitive Science, 5, p. 1-30, 1981.

[LAWLER 87] R. W. Lawler
" Learning Environments : now, then, and someday "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 1-25, 1987.

[LE CORRE 87] Yves Le Corre
« Les concepts scientifiques et technologiques »
Rapport O. C. D. E. (CERI), 1987.

[LE MEUR 90] André Le Meur
" Hypertext and more. ARGOS : a new approach to learning by seeking "
Applica 90, Lille, Septembre 1990.

[LEBLANC 88] M. D. LeBlanc
" Instructional tools for algebra word problems "
in ITS-88, Montréal, p. 238-242, 1988.

[LEGGETT & Al. 90] J. J. Leggett, L. Schnase, C. J. Kacmar
" Hypertext for learning "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[LEHNING 88] H. Lehning
" Mathematics of Tomorrow "
in Computers in Education Elsevier, IFIP p. 615-619, 1988.

[LENAT 83] Douglas B. Lenat
" The role of heuristics in learning by discovery : three case studies "
in J. G. Carbonell, R. Michalski & T. Mitchell *" Machine Learning, an A. I. approach "*
Springer Verlag, p. 246-306, 1983.

[LEPPER & CHABAY 88] M. R. Lepper, R. W. Chabay
" Socializing the Intelligent Tutor : Bringing Empathy to Computer Tutors "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 242-257, 1988.

[LESGOLD 87] Alan M. Lesgold
« Les technologies de l'information et apprentissages de base : Principaux problèmes et perspectives d'avenir »
Rapport O. C. D. E. (CERI), 1987.

[LEVINE & POMEROL 89] Pierre Lévine et Jean-Charles Pomerol
« Systèmes interactifs d'aide à la décision »
Hermès, 1989.

[LEWIS & Al. 87] M. W. Lewis, R. Milson, J. R. Anderson
" The TEACHER'S APPRENTICE : Designing an Intelligent Authoring System for High School Mathematics "

in G. P. Kearsley (ed.) " *Artificial Intelligence and Instruction. Applications and Methods* ", Addison-Wesley, p. 269-301, 1987.

[LIEBERMAN 87] H. Lieberman
" *An example-based environment for beginning programmers* "
in Lawler and Yazdani " *Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems* " Ablex, p. 135-151, 1987.

[LITTMAN & SOLOWAY 88] D. Littman and E. Soloway
" *Evaluating ITS : the cognitive science perspective* "
in M. C. Polson and J. J. Richardson (eds) " *Foundations of ITS* "
Hillsdale, LEA, p. 209-242, 1988.

[LOI 82] Maurice Loi
« *Rigueur et ambiguïté* »
in « *Penser les Mathématiques* » Seuil, 1982.

[LOSER & KURTZ 89] M. Loser, B. Kurtz
" *Quadratic Grapher : An ITS for graphing quadratic equations* "
in E. Maurer (ed.) " *Computer-Assisted Learning* ", Springer Verlag, p. 346-358, 1989.

[LOOI 88] C. K. Looi
" *Representation of task and programming knowledge in I. T. S. for programming* "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 17-20, 1988.

[MADAULE & Al. 87] F. Madaule, P. Barril, B. de la Passardière, F. le Clavez, M. M. Poc, M. Urtasun « *Systèmes d'Enseignement Assisté par Ordinateur : étude comparative* »
TSI, vol. 6, n° 1, 1987.

[MARCHIONINI 90] Gary Marchionini
" *Evaluating Hypermedia-Based Learning* "
in " *Designing Hypermedia for Learning* ", NATO ASI Series, vol. F67, Springer, 1990.

[MARINA-MEDIAVILLA] A. M. Marina-Médiavilla
« *METHODES : Apprentissage méthodique du commentaire composé* »
Editions Modernes MEDIA.

[MARTIN & FATEMAN 71] W. A. Martin, R. J. Fateman
" *The MACSYMA System* "
Proceedings SS-SAM, Los Angeles, p. 59-75, 1971.

[MATHIEU & THOMAS 85] Mathieu Jacques et Raymond Thomas
« *Manuel de Psychologie* »
Vigot, 1985.

[MATZ 82] M. Matz
" *Towards a process model for high school algebra errors* "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 25-50, 1982.

[MAURER 87] S. B. Maurer
" *New Knowledge about Errors and New Views about Learners : What they Mean to Educators and More Educators Would like to Know* "

in Schoenfeld (ed.) " *Cognitive Science and Mathematics Education* " LEA, p. 165-187, 1987.

[MAYER 83] Richard E. Mayer
" *Thinking, Problem Solving, Cognition* "
W. H. Freeman and Company, 1983.

[MAYES & Al. 90] T. Mayes, M. Kibby, T. Anderson
" *Learning about Learning from Hypertext* "
in " *Designing Hypermedia for Learning* ", NATO ASI Series, vol. F67, Springer, 1990.

[McARTHUR & Al. 88] D. McArthur, C. Burdof, T. Orsmeth, A. Robyn, C. Stasz
" *Multiple representations of mathematical reasoning* "
in ITS-88, Montréal, p. 430-434, 1988.

[McCALLA & GREER 88] G. I. McCalla, J. E. Greer
" *Intelligent Advising in problem solving domains : the SCENT-3 architecture* "
in ITS-88, Montréal, p. 124-131, 1988.

[McCOY CARVER 86] Sharon McCoy Carver
" *Transfer of Logo Debugging Skill : Analysis, Instruction and Assesment* "
PhD-Thesis, Carnegie-Mellon University, september 1986.

[McDONALD 81] J. McDonald
" *The EXCHECK CAI system* "
in P. Suppes (ed.), " *University-level CAI at Standford : 1968-1980* " Standford University, California, 1981.

[McKENZIE 90] I. Scott MacKenzie
" *Courseware evaluation : where's the intelligence?* "
Journal of Computer Assisted Learning, 6, p. 273-285, 1990.

[MEILLEUR & MITCHELL 89] D. Meilleur, P. D. Mitchell
" *Concept clarification and visual representation for a knowledge engineering advisory tool* "
Proceedings of the sixth Canadian Symposium on Instructional Technology p 173-177, 1989

[MENDELSOHN 88] P. Mendelsohn
« *Les activités de programmation chez l'enfant* »
Revue T. S. I., Vol. 7, n° 1, février 88.

[MERIALDO 79] B. Mérialdo
« *Représentation des ensembles en démonstration automatique* »
Thèse de 3ème cycle, Paris VI, 1979.

[MERRI 90] Maryvonne Merri
« *Les apports d'un modèle d'acquisition à la conception d'un système d'EIAO de la notion de proportionnalité* »
Applica 90, Lille, 1990.

[MEYROWITZ 89] Norman Meyrowitz
" *The Missing Link : Why We're All Doing Hypertext Wrong* "
in The Society of Text, Edward Barett (ed.), MIT Press, p. 107-114, 1989.

[MICHARD 85] Alain Michard
« *Reconnaissance et génération de plans d'actions : Application à la réalisation de*

systèmes auto-explicatifs »
Cognitiva, Paris, p. 225-231, juin 1985.

[MIDORO & OLIMPO 90] V. Midoro, G. Olimpo
" Educational Systems based on multimedia databases of learning material "
in The Seventh International Conference on Technology and Education CEP Consultants LTD, p. 297-299, 1990.

[MIDORO & Al. 90] V. Midoro, G. Olimpo, D. Persico, L. Sarti
" Multimedia navigable systems and Artificial Intelligence "
International Conference on ARCE, Tokyo, p. 155-160, juillet 1990.

[MILLER 88] J. R. Miller
" The Role of Human-Computer Interaction in ITS "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 143-190, 1988.

[MITCHELL & Al. 83] T. M. Mitchell, P. E. Utgoff, R. Banerji
" Learning by experimentation : acquiring and refinig problem-solving heuristics "
in J. G. Carbonell, R. Michalski & T. Mitchell *" Machine Learning, an A. I. approach "*
Springer Verlag, p. 163-190, 1983.

[MITRE 76] Mitre Corporation
" An overview of the TICCIT program "
Report M76-44, Washington, Mitre Corporation, 1976.

[MOBUS 90] Claus Möbus
" The Relevance of Computational Models of Knowledge Acquisition for the Design of Helps in the Problem Solving Monitor ABSYNT "
International Conference on ARCE, Tokyo, p. 57-64, juillet 1990.

[MORI & Al. 90] E. Mori, A. Takeuchi, S. Otsuki
" Thinking Tool : improving student's reasoning abilities "
International Conference on ARCE, Tokyo, p. 49-56, juillet 1990.

[MOSES 71a] J. Moses
" Symbolic Integration : the stormy decade "
Communications ACM 14, p. 548-560, 1971.

[MOSES 71b] J. Moses
" Algebraic simplification : a guide for the perplexed "
Communications ACM 14, p. 527-537, 1971.

[MONTES 88] Antonio Montès
" Use of specialized languages in Education "
in Computers in Education Elsevier, IFIP p. 219-223, 1988.

[MOUSTAFIADES 90] Jeannine Moustafiadès
« Formation au diagnostic technique : l'apport de l'I. A. »
Masson, 1990.

[MOYSE 89] R. Moyse
" Knowledge Negotiation Implies Multiple Viewpoints "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 140-149, 1989.

[NATHAN & Al. 89] M. J. Nathan, P. Johl, W. Kintsch, C. Lewis
" An unintelligent tutoring system for solving Word Algebra problems "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 169-176, 1989.

[NELSON 70] T. Nelson
" No More Teachers' Dirty Looks "
Computer Decisions, September, p. 16-23, 1970.

[NELSON 81] T. Nelson
" Literary Machines "
1981 (version hypertexte en 1987).

[NELSON 89] Theodor Holm Nelson
" Hyperwelcome "
Hypermedia, Taylor Graham vol. 1, 1, 1989.

[NEVES 78] D. M. Neves
" Learning procedures from examples "
Unpublished doctoral dissertation, CMU, 1978.

[NEWELL & SIMON 72] A. Newell, H. A. Simon
" Human Problem Solving "
Englewood Cliffs, NJ :Prentice-Hall, 1972.

[NEWMAN 89] Denis Newman
" Is a Student Model Necessary? Apprenticeship as a Model for ITS "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 177-184, 1989.

[NICAUD 87] J. F. Nicaud
« Aplusix : un système expert en résolution pédagogique d'exercices d'algèbre »
Thèse d'Université Paris XI, 1987.

[NICAUD 88a] J. F. Nicaud
« APLUSIX : Un système expert de résolution pédagogique d'exercices d'algèbre »
Bulletin de liaison n° 1, IREM de Toulouse, p. 41-74, février 1988.

[NICAUD 88b] J. F. Nicaud
" APLUSIX : a learning environment on algebraic manipulations in order to develop the student's ability in searching "
Summer University Le Mans 1988.

[NICAUD & VIVET 88] J. F. Nicaud et M. Vivet
« Les tuteurs intelligents. Réalisations et tendances de recherches »
TSI 7 (1988) n° 1, 21-45

[NICHOL 88] Jon Nichol
" Knowledge Based Tools (shells) and children "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 292-295, 1988.

[NICOLSON 88] R. I. Nicolson
" The SUMMIT Intelligent Arithmetic Tutor "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 348-351, 1988.

[NIELSEN 90] J. Nielsen
" Evaluating Hypertext Usability "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[NWANA & COXHEAD 88] H. S. Nwana, P. Coxhead
" Towards an Intelligent Tutoring System for Fractions "
in ITS-88, Montréal, p. 403-408, 1988.

[O'SHEA 82a] Tim O'Shea
" A self-improving quadratic tutor "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 309-336, 1982.

[O'SHEA 82b] Tim O'Shea
" Intelligent systems in education "
in D. Michie, *" Introductory Readings in Expert Systems "* Gordon and Breach science publishers, p. 147-176, 1982.

[O'SHEA 89] Tim O'Shea
" Magnets, Martians and Microworlds : Learning with and learning by OOPS "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 193, 1989.

[O'SHEA & Al. 84] T. O'Shea, R. Bornat, B. du Boulay, M. Eisenstadt, I. Page
" Tools for creating intelligent computer tutors "
in *" Artificial and Human Intelligence "* North Holland, 1984.

[O'SHEA & Al. 88] T. O'Shea, R. Evertsz, S. Hennessy, A. Floyd, M. Fox, M. Elsom-Cook
" Design Choices for an intelligent arithmetic tutor "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 257-276, 1988.

[O'SHEA & SELF 83] Tim O'Shea and John Self
" Learning and Teaching with computers "
Artificial Intelligence in Education Harvester Press, 1983.

[OCDE 87]
« Technologies de l'information et apprentissages de base : Lecture, écriture, sciences et mathématiques »
OCDE, 1987.

[OHLSSON 86] Stellan Ohlsson
" Some principles of Intelligent Tutoring "
(Instructional Science 14, p. 293-326, 1986.) in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "*, Ablex, p. 203-238, 1987.

[OHLSSON 87] Stellan Ohlsson
" Sense and Reference in the Design of Interactive Illustrations for Rational Numbers "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "*, Ablex, p. 307-344, 1987.

[OHLSSON & LANGLEY 88] Stellan Ohlsson, Pat Langley
" Psychological Evaluation of Path Hypotheses in Cognitive Diagnosis "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 42-62, 1988.

[OLIVER & ZUKERMAN 90] J. Oliver, I. Zukerman
" DISSOLVE : an algebra expert for an ITS "
International Conference on ARCE, Tokyo, p. 149-154, juillet 1990.

[OR-BACH & BAR-ON 89] R. Or-Bach and E. Bar-On
" PROBIT - Developing a diagnostic intelligent tutor "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 185-192, 1989.

[PALIES 88a] Odile Paliès
« Méta-connaissances pour la modélisation de l'élève. Contribution au diagnostic cognitif par système expert »
Thèse Paris VI, mars 1988.

[PALIES 88b] Odile Paliès
« Diagnostic du fonctionnement cognitif et apprentissage symbolique »
in ITS-88, Montréal, p. 201-206, 1988.

[PALIES & Al. 88] O. Paliès, M. Caillot, E. Cauzinille, J. L. Laurière, J. Mathieu
" Student Modelling by a knowledge-based system "
Summer University Le Mans 1988.

[PAPERT 80] Seymour Papert
" Mindstorms, Children, computers and powerful ideas "
(*« Jaillisement de l'esprit »*), Flammarion, 1980.

[PAPERT 85] Seymour Papert
" Computer Criticims vs. Technocentric Thinking "
LOGO 85, Theoretical Papers, MIT, Cambridge, July 1985.

[PAPERT 87] Seymour Papert
" Microworlds : Transforming Education "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 79-94, 1987.

[PAQUETTE 88] Gilbert Paquette
« Le développement d'outils intelligents d'apprentissage pour le traitement des connaissances »
in ITS-88, Montréal, p. 75-81, 1988.

[PASQUIER & COLLAUD 87] Jacques Pasquier-Boltuk & Gérald Collaud
" The Woven Electronic Book System (WEBS) : The Enduser Model and Interface "
Institut pour l'automation et la recherche opérationnelle Université de Fribourg/Suisse, 1987.

[PASTRE 76] D. Pastre
« Démonstration automatique de théorèmes en théorie des ensembles »
Thèse de 3ème cycle, Paris VI, 1976.

[PASTRE 78] D. Pastre
« Observation du mathématicien : Aide à l'enseignement et à la démonstration automatique de théorèmes. »
Educational Studies in Mathematics, vol. 9, 1978, p. 259-285

[PASTRE 84] Dominique Pastre
« MUSCADET : un système de démonstration automatique de théorèmes utilisant

connaissances et métaconnaissances en mathématiques »
Thèse de doctorat d'Etat, Paris VI, Novembre 1984.

[PASTRE 85] Dominique Pastre
« Expérimentation d'une expertise mise au point par l'observation du comportement de l'homme »
Cognitiva, Paris, p. 245-250, juin 1985.

[PEA & KURLAND 84] R. D. Pea and D. M. Kurland
" Logo programming and the development of planning skills "
Technical Report n° 16, New York, NY : Center for Children and Technology, Bank Street College of Education

[PEA & Al. 85] R. D. Pea, D. M. Kurland, J. Hawkins
" Logo and the development of Thinking Skills "
in M. Chen & W. Paisley, *" Children and Microcomputers. Research on the Newest Medium "*, London, Sage Publications, p. 193-212, 1985.

[PEA 87] Roy D. Pea
" Cognitive Technologies for Mathematics Education "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 89-122, 1987.

[PEACHAM & WOOLLIAMS 90] J. Peacham, P. Woolliams
" Marrying Hypertext with Free Text Retrieval to provide learning materials with improved pupil / student satisfaction "
in The Seventh International Conference on Technology and Education CEP Consultants LTD, p. 119-121, 1990.

[PERKINS & SIMMONS 88] D. N. Perkins, R. Simmons
" Patterns of Misunderstanding : An Integrative Model for Science, Math and Programming "
Review of Educational Research, vol. 58, n°3, p. 303-326, 1988.

[PIAGET] Jean Piaget
« La genèse du nombre »
Collection La Pléiade

[PIROLLI & RUSSELL 88] P. Pirolli and D. Russell
" Towards theory and technology for the design of ITS "
in ITS-88, Montréal, p. 350-356, 1988.

[PITRAT 66] Jacques Pitrat
« Réalisation de programmes de démonstration de théorèmes utilisant des méthodes heuristiques »
Thèse d'Etat, Paris, 1966.

[PITRAT 84] Jacques Pitrat
" An intelligent system can and must use declarative knowledge efficiently "
in Artificial and Human Intelligence Elsevier, 1984.

[PITRAT 86] J. Pitrat
« Connaissances et méta-connaissances »
in Intelligence des Mécanismes et mécanismes de l'Intelligence Fondation Diderot, Fayard (ed : J. L. Lemoigne), p. 75-113, 1986.

[PITRAT 90] J. Pitrat
« Métaconnaissances : futur de l'intelligence artificielle »
Hermès, 1990.

[PLAISANT-SCHWENN 89] Catherine Plaisant-Schween
« L'interface utilisateur d'un système Hypermedia. L'exemple Hyperties »
Bulletin du CID n°4, mai 1989.

[POLYA 57] G. Polya
« Comment poser et résoudre un problème »
Dunod Editeur, Paris, 1957.

[PORTER & KIBLER 86] B. W. Porter, D. F. Kibler
" A method for Learning Problem-Solving Heuristics "
Machine Learning, vol. 1, p. 249-286, 1986.

[PUTMAN 85] R. T. Putman
" Teacher thoughts and actions in live and simulated tutoring of addition "
Doctoral dissertation, Standford University, 1985.

[PY 89] Dominique Py
" MENTONIEZH : an Intelligent Tutoring System in geometry "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 202-209, 1989.

[QUIGLEY 89a] M. T. Quigley
" EMMA : an intelligent tutor "
Proceedings of the sixth Canadian Symposium on Instructional Technology p. 95-98, 1989.

[QUIGLEY 89b] M. Quigley
" Computor Tutoring of Linear Equations "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 210-217, 1989.

[REGOURD 85] J. P. Regourd
« Apprentissage Autonome et Intelligence Artificielle »
Thèse de 3ème cycle, Université Paris VI et LITP France, décembre 1985.

[REGOURD 88] J. P. Regourd
« Définition et implémentation d'un niveau méta en E. I. A. O. »
in ITS-88, Montréal, p. 321-325, 1988.

[REIMANN 88] P. Reimann
" Learning by discovery in a microworld for geometrical optics "
Summer University Le Mans 1988.

[REISER & Al. 89] B. J. Reier, M. Ranney, M. C. Lovett, D. V. Kimberg
" Facilitating Students' Reasoning with Causal Explanations and Visual Represenations "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 228-235, 1989.

[RESNICK 82] Lauren B. Resnick
" Syntax and Semantics in Learning to Subtract "

in Carpenter, Moser, Romberg (eds), *" Addition and subtraction : a cognitive perspective "*, LEA, p. 137-155, 1982.

[RESNICK 83] Lauren B. Resnick
" Beyond Error Analysis : The Role of Understanding in Elementary School Arithmetic "
LRDC, Pittsburgh, 1983.

[RICHARDSON 88] J. J. Richardson
" Directions for Research and Applications "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 243-252, 1988.

[RICHER & CLANCEY 87] M. H. Richer, W. J. Clancey
" GUIDON-WATCH : a Graphic Interface for Viewing a Knowledge-Based System "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "*, Ablex, p. 373-412, 1987.

[RIDGWAY 88] Jim Ridgway
" Of course ICAI is impossible... worse though, it might be seditious "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 28-48, 1988.

[ROBERT 85] Frédéric Robert
« Un exemple d'environnement déclaratif en LOGO »
Document de travail I. N. R. P./D. P. 5, mars 1985.

[ROBERT 86] Frédéric Robert
« LOGOGRAM : Un environnement déclaratif en Logo, pour construction de "grammaires" destinées à un générateur ou un analyseur »
I. N. R. P./D. P. 5, décembre 1986.

[ROBERT & Al. 87] A. Robert, J. Rogalski et R. Samurcay
« Enseigner des méthodes »
Cahier de didactique des Mathématiques n° 15 IREM Paris VII, mars 1987

[ROBINET 86] J. Robinet
« Les réels : quels modèles en ont les élèves »
Cahier de didactique des Mathématiques n° 15 IREM Paris VII, 1986

[ROBINSON 65] J. A. Robinson
" A machine oriented logic based on the resolution principle "
J. ACM, vol. 12, p. 23-41, 1965.

[ROGALSKI 87] J. Rogalski
« Epistémologie génétique et didactique : pour une théorie de l'acquisition des connaissances complexes »
Colloque de la Société Française de Psychologie : les apprentissages - perspectives actuelles, Saint-Denis, janvier 1987.

[ROHR & TAUBER 85] G. Rohr, M. J. Tauber
" Representational frameworks and models for human-computer interface "
in *" User Centered System Design : New perspectives on Human-Computer Interaction "*, 1985.

[ROMISZOWSKI 90] Alexander J. Romiszowski
" The Hypertext / Hypermedia Solution - But what exactly is the Problem? "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[ROSNICK & CLEMENT 80] P. Rosnick, J. Clement
" Learning Without Understanding : The Effect of Tutoring Strategies on Algebra Misconceptions "
Journal of Mathematical Behavior, vol. 3, n°1, p. 3-27, 1980

[ROSS 87] P. Ross
" Intelligent tutoring systems "
Journal of Computer Assisted Learning, 3, p. 194-203, 1987.

[ROSS & HOWE 81] Peter Ross and Jim Howe
" Teaching mathematics through programming : ten years on "
in R. Lewis & D. Tagg (eds) *" Computers in Education "* North Holland, 1981.

[ROSS & LEWIS 87] Peter Ross and John Lewis
" Plan recognition for intelligent tutoring systems "
IFIP / TC3 FRASCATI North-Holland, 1987.

[ROUET 90] J. F. Rouet
« Lecture et Orientation Sémantique dans les Documents Non-Linéaires »
Applica 90, Lille, Septembre 1990.

[ROUSSET 83] M. C. Rousset
« TANGO : moteur d'inférence pour une classe de S. E. avec variables »
Thèse de 3ème cycle, Paris XI, 1983.

[ROWE 76] Neil Rowe
Grammar as a programming language LOGO-MEMO 39 (A. I. mémo 391), M. I. T., 1976.

[ROWE 88] Neil C. Rowe
" Artificial Intelligence through Prolog "
Prentice-Hall, 1988.

[RUBENS 89] P. Rubens
" Online Information, Hypermedia, and the Idea of Literacy "
in The Society of Text, Edward Barett (ed.), MIT Press, p. 3-21, 1989.

[RUSSELL 90] D. M. Russell
" Alexandria : a learning resources management architecture "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[SACK 88] Warren Sack
" Finding errors by overlooking them "
in ITS-88, Montréal, p. 395-402, 1988.

[SAVOY 87] Jacques Savoy
« Le livre électronique EBOOK3 »
Colloque E. A. O. 87 Institut pour l'automation et la recherche opérationnelle Université de Fribourg/Suisse, 1987.

[SCHANK & EDELSON 89] R. C. Schank and D. J. Edelson
" Discovery Systems "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 236-237, 1989.

[SCHOENFELD 87a] Alan H. Schoenfeld
" Cognitive Science and Mathematics Education : An Overview "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 1-31, 1987.

[SCHOENFELD 87b] Alan H. Schoenfeld
" What's All the Fuss About Metacognition "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 189-215, 1987.

[SCHOFIELD & EVANS-RHODES 89] J. W. Schofield and D. Evans-Rhodes
" AI in the Classrom : The Impact of a Computer-Based Tutor on Teachers and Students "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 238-243, 1989.

[SCHOFIELD & Al. 90] J. Schofield, D. Evans-Rhodes, B. R. Huber
" AI in the classroom : The Impact of a Computer-Based Tutor on Teachers and Students "
Social Science Computer Review (in press), 1990.

[SCHULZE 89] K. G. Schulze
" PRECEPTOR : an algebraic word problem tutor "
Proceedings of the sixth Canadian Symposium on Instructional Technology p 63-67, 1989.

[SCHWOB & BLONDEL 89] M. Schwob, F. M. Blondel
« Comment les élèves ne résolvent pas les problèmes de chimie »
in Proceedings of XIèmes Journées Internationales sur l'Education Scientifique Chamonix, p. 311-316, 1989.

[SELF 74] J. Self
" Student models in Computer-Aided Instruction "
Int. J. Man-Machine Studies, 6, p. 261-276, 1974.

[SELF 85a] John Self
" Microcomputers in Education : a critical appraisal of educational software "
Harvester Press, 1985.

[SELF 85b] John Self
" A perspective on intelligent computer-assisted learning "
Journal of Computer Assisted Learning 1. p. 159-166, 1985

[SELF 87] J. A. Self
" Student's models : what use are they? "
in P. Ercoli and R. Lewis (eds), *" Artificial Tools in Education "* Amsterdam, North Holland, 1988.

[SELF 88] J. A. Self
" Bypassing the intractable problem of student modelling "
in ITS-88, Montréal, p. 18-24, 1988.

[SELF 89] John Self
" The Case for Formalising Student Models (and ITS generally) "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 244, 1989.

[SELKER 89] Ted Selker
" COgnitive Adaptive Computer Help (COACH) "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 245-251, 1989.

[SENDOV & DICHEVA 88] B. Sendov and D. Dicheva
" A mathematical laboratory in Logo style "
in Computers in Education Elsevier, IFIP p. 213-217, 1988.

[SERRES 68] Michel Serres
« HERMES I : la communication »
Editions de Minuit, 1968.

[SHAPIRO & STERLING 86] Leon Sterling et Ehud Shapiro
" The Art of Prolog "
M. I. T. Press, 1986.

[SHARPLES 85] Mike Sharples
" Cognition, Computers and Creative Writing "
Ellis Horwood series, 1985.

[SHNEIDERMAN 82] Ben Shneiderman
" The future of interactive systems and the emergence of direct manipulation "
Behavior and Information Technology, vol. 1, n°3, p. 237-256, 1982.

[SHNEIDERMAN 83] Ben Shneiderman
" Direct Manipulation : a Step Beyond Programming Languages "
Computer, vol. 16, n°8, p. 57-69, 1983.

[SHNEIDERMAN 89] Ben Shneiderman
" Reflections on Authoring, Editing and Managing Hypertext "
in The Society of Text, Edward Barett (ed.), MIT Press, p. 115-131, 1989.

[SHNEIDERMAN & KEARSLEY 89] B. Shneiderman, G. Kearsley
" Hypertext Hands-On! "
Addison-Wesley, MA, 1989.

[SILVER 86] B. Silver
" Precondition Analysis : Learning Control Information "
in Machine Learning : An Artificial Intelligence Approach, Palo Alto, p. 647-670, 1986.

[SILVER 87] E. A. Silver
" Foundations of Cognitive Theory and Research for Mathematics Problem-Solving Instruction "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 33-60, 1987.

[SINGLEY 88] M. K. Singley
" The relationship between operator selection and application in mathematics problem solving skill "
in ITS-88, Montréal, p. 365-371, 1988.

[SINGLEY & Al. 89] M. K. Singley, J. R. Anderson, J. S. Gevins, D. Hoffman
" The Algebra Word Problem Tutor "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 267-275, 1989.

[SKINNER 68] B. F. Skinner
" The technology of Teaching "
New-York : Appleton-Century-Crofts, 1968.

[SLAGLE 61] J. R. Slagle
" A heuristic program that solves symbolic integration problems in freshman calculus,

Symbolic Automatic INTegrator "
Thèse MIT, Lincoln Laboratory, 1961.

[SLATIN 88] J. M. Slatin
" *Hypertext and the Teaching of Writing* "
in E. Barrett (ed) " *Text, ConText, and HyperText* ", p. 111-129, 1988.

[SLEEMAN 82] D. Sleeman
" *Assessing aspects of competence in basic algebra* "
in Intelligent Tutoring systems (D. Sleeman and J. S. Brown, eds), Academic Press, p. 185-199, 1982.

[SLEEMAN 83] D. Sleeman
" *Inferring Student Models for Intelligent Computer-Aided Instruction* "
in J. G. Carbonell, R. Michalski & T. Mitchell " *Machine Learning, an A. I. approach* "
Springer Verlag, p. 483-510, 1983.

[SLEEMAN 87a] D. Sleeman
" *Micro-SEARCH : A 'Shell' for Building Systems to Help Students Solve Non-deterministic Tasks* "
in Kearsley " *Artificial Intelligence and Instruction, applications and methods* ", Addison-Wesley, 1987.

[SLEEMAN 87b] D. Sleeman
" *PIXIE : A Shell for developing Inteligent Tutoring Systems* "
in Lawler and Yazdani " *Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems* ", Ablex, p. 239-263, 1987.

[SLEEMAN & Al. 89] D. Sleeman, A. E. Kelly, R. Martinak, R. D. Ward, J. L. Moore
" *Studies of Diagnosis and Remediation with High School Algebra Students* "
Cognitive Science, 13, p. 551-568, 1989.

[SLEEMAN & BROWN 82] D. Sleeman and J. S. Brown
" *Intelligent Tutoring systems* "
Academic Press, 1982.

[SMITH 86] R. B. Smith
" *The Alternate Reality Kit : An Animated Environment for Creating Interactive Simulations* "
Proceedings of the 2nd IEEE Computer Society Workshop on Visual Languages, Dallas, p. 99-106, 1986.

[SOLOMON 85] Cynthia Solomon
" *Computer Environments for children : A reflection on Theories of Learning and Education* "
MIT Press, 1985.

[SOLOWAY & Al. 83a] E. M. Soloway, E. Rubin, B. P. Woolf, J. Bonar, W. L. Johnson
" *MENO-II an AI-based programming tutor* "
Journal of Computer-BAsed Instruction, vol. 10, n° 1, p. 20-34, 1983.

[SOLOWAY & Al. 83b] E. Soloway, K. Ehrlich, V. Abbot
" *Transfer Effects from Programming to Algebra Word Problems : A Preliminary Study* "
Technical Report 257, Yale University, november 1983

[STEGMAIER 88] E.-L. Stegmaier
" CUM the computer as a means of instruction "
Summer University Le Mans 1988.

[STUMPF & Al. 88] M. stumpf, K. Opwis, H. Spada
" Knowledge Acquisition in a Microworld for Elastic Impacts : the DiBi System "
Summer University Le Mans 1988.

[SUPPES 79] P. Suppes
" Current trands in CAI "
Advances in Computers, Vol. 18, p. 173-229, Academic Press, 1979.

[SKWARECKI 88] Edward J. Skwarecki
" Improving the engineering of model-tracing diagnosis "
in ITS-88, Montréal, p. 215-221, 1988.

[SWAN 89] Karen Swan
" The Teaching and Learning of Problem Solving through LOGO programming "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 281-290, 1989.

[TAKEUCHI & OTSUKI 87] A. Takeuchi and S. Otsuki
" A study on student models and learner machine interaction "
IFIP / TC3 FRASCATI North-Holland, 1987.

[TANG & Al. 90] H. Tang, R. Barden, C. Clifton
" A new learning environment based on hypertext adn ITS techniques "
International Conference on ARCE, Tokyo, p. 39-47, juillet 1990.

[TEXIER 88] Alain Texier
« Des ailes pour la tortue »
Document CNDP, 1988.

[THOMPSON 87] P. W. Thompson
" Mathematical Microworlds and Intelligent Computer-Aided Instruction "
in G. P. Kearsley (ed.) *" Artificial Intelligence and Instruction. Applications and Methods "*, Addison-Wesley, p. 83-109, 1987.

[TIJUS & Al. 90] C. A. Tijus, P. Chevalier, M. Fargue, J. Suchard
« Principes pour un atelier de conception de logiciels EAO : l'exemple du générateur de cas ArbraCas »
Document CNAM, 1990.

[TWIDALE 89] Michael Twidale
" Intermediate representations for student error diagnosis and support "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 298-306, 1989.

[VAN DER VEER & BEISHUIZEN 84] G. van der Veer & J. Beishuizen
" The computer in the classroom "
in *" Readings on cognitive ergonomics, mind and computers "*, Berlin, Springer Verlag, p. 170-179, 1984.

[VANLEHN 81] Kurt VanLehn
" On the representation of procedures in Repair Theory "

in The Development of Mathematical Thinking, H. P. ginsberg (Ed.), Academic Press, 1981.

[VANLEHN 87] Kurt VanLehn
" Learning one subprocedure per lesson "
Artificial Intelligence, 31, p1-40, 1987.

[VANLEHN 88a] K. VanLehn
" Toward a theory of Impasse-Driven Learning "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 19-41, 1988.

[VANLEHN 88b] K. VanLehn
" Student Modeling "
in M. C. Polson and J. J. Richardson (eds) *" Foundations of ITS "*, Hillsdale, LEA, p. 55-78, 1988.

[VANLEHN 90] Kurt VanLehn
" Two pseudo-students : application of machine learning to formative evaluation "
International Conference on ARCE, Tokyo, p. 181-190, juillet 1990.

[VANLEHN & BROWN 80] K. Van Lehn and J. S. Brown
" Planning nets : a representation for formalizing analogies and semantic models of procedural skills "
in Aptitude, learning and instruction (Snow, Frederico, Montague, eds) vol 2, LEA, p. 96-137, 1980.

[VERGNAUD & CORTES 87] G. Vergnaud et A. Cortès (P. Favre-Artigue)
« Apprentissage de l'algèbre par des élèves faibles : 4ème et 3ème de LEP »
Colloque de la Société Française de Psychologie : les apprentissages - perspectives actuelles, Saint-Denis, janvier 1987.

[VISETTI 86] Y. M. Visetti
« Contribution à la modélisation de l'interlocuteur en EAO »
Thèse de Doctorat, Paris VI, décembre 86.

[VIVET 83] Martial Vivet
« Un exemple d'utilisation de méta-règles : la sélection des plans dans CAMELIA »
Publication GR22-CNRS, décembre 1983.

[VIVET 84a] Martial Vivet
« Expertise mathématique et informatique : CAMELIA un logiciel pour raisonner et calculer »
Thèse de doctorat d'Etat, Université Paris VI, Juin 1984,

[VIVET 84b] Martial Vivet
« Regards différents autour de CAMELIA »
Colloque Intelligence Artificielle Aix-en-Provence, Publication du GR22, Septembre 84.

[VIVET 87] Martial Vivet
« Systèmes Experts pour enseigner méta-connaissances et explications »
Cognitiva 87, février 1987.

[VIVET 88] Martial Vivet
" Knowledge based tutors : towards the design of a shell "
International Journal of Educational Research, vol. 12, 8, p. 839-850, 1988.

[VIVET & Al. 88] M. Vivet, M. Futtersack, J. M. Labat
« Métaconnaissance dans les tuteurs intelligents »
in ITS-88, Montréal, p. 430-434, 1988.

[WACHSMUTH 88] Ipke Wachsmuth
" Modeling the Knowledge Base of Mathematics Learners : Situation-Specific and Situation-Nonspecific Knowledge "
in H. Mandl, A. Lesgold (eds), *" Learning Issues for ITS "* Springer-Verlag, p. 63-79, 1988.

[WAERN 84] Yvonne Waern
" On the implications of user's prior knowledge for Human-computer interaction "
in *" Readings on cognitive ergonomics, mind and computers "*, Berlin, Springer Verlag, p. 144-159, 1984.

[WALSH 88] Toby Walsh
" PLATO : Predicate Logic Advisory TOol "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 9-12, 1988.

[WANG 60] H. Wang
" Towards mechanical mathematics "
IBM J. Res. Develop. 4, p. 2-22, 1960.

[WARD & Al. 88] R. D. Ward, I. K. Kirby, J. B. Smith, J. R. Cramond, D. Sleeman, K. Lowe
" Building an ITS within an expert system shell "
in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 2, p. 13-16, 1988.

[WEIDENFELD 87] Gérard Weidenfeld
" SEVE "
Proceedings of the Fifth International Joint Conference, Pittsburgh, 1987

[WENGER 87] Etienne Wenger
" Artificial Intelligence and Tutoring Systems "
Morgan Kaufmann, 1987.

[WENGER 87] Ronald H. Wenger
" Cognitive Science and Algebra Learning "
in Schoenfeld (ed.) *" Cognitive Science and Mathematics Education "* LEA, p. 217-251, 1987.

[WERTZ 85] H. Wertz
« Intelligence Artificielle, application à l'analyse de programmes »
Masson, 1985.

[WHALLEY 90] P. Whalley
" Models of Hypertext Structure and Learning "
in *" Designing Hypermedia for Learning "*, NATO ASI Series, vol. F67, Springer, 1990.

[WHITE & FREDERIKSEN 87] B. Y. White, J. R. Frederiksen
" Qualitative Models and Intelligent Learning Environments "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 281-305, 1987.

[WINANS & Al. 88] R. T. Winans, E. T. Whitaker, R. D. Bonnell
" Theories of Learning in Intelligent Computer-Aided Instruction "

in The Fifth International Conference on Technology and Education CEP Consultants Ltd, vol 1, p 86-89, 1988.

[WINKELS & Al. 88] R. Winkels, J. Breuker, J. Sandberg
" Didactic Discourse in Intelligent Help Systems "
in ITS-88, Montréal, p. 279-285, 1988.

[WOLFRAM 88] S. Wolfram
" Mathematica : A system for doing Mathematics by Computer "
Redwood City, Addison-Wesley, 1988.

[WOODROFFE 88] M. R. Woodroffe
" Plan recognition and intelligent tutoring systems "
in J. Self (ed.) *" Artificial and Human Learning "*, Chapman and Hall Computing, p. 212-225, 1988.

[WOOLF & McDONALD 84] Beverly Woolf and David Mc Donald
" Building a Computer Tutor : Design Issues "
IEEE, 1984.

[YANKELOVITCH & Al. 85] N. Yankelovitch, N. Meyrowitz, A. Van Dam
" Reading and Writing the Electronic Book "
Computer, Volume 18, Number 10, p. 15-30, October 85.

[YANKELOVITCH 85] N. Yankelovitch, G. P. Landow, D. Cody
" Creating Hypermedia Materials for English Literature Students "
IRIS, Brown University, October 85.

[YAZDANI 82] M. Yazdani
" New Horizon in Educational Computing "
Ellis Horwood, 1982.

[YAZDANI 87] M. Yazdani
" ITS : an Overview "
in Lawler and Yazdani *" Artificial Intelligence and Education, Vol 1 : Learning Environments & Tutoring Systems "* Ablex, p. 183-201, 1987.

[YAZDANI 89] Masoud Yazdani
" Second Generation ICAI Systems "
in Proceedings of the 4th International Conference on AI and Education Amsterdam, IOS, p. 338-339, 1989.

[YERUSHALMY 88] Michal Yerushalmy
" Computer data generated by geometry students : criteria for an appropriate information. "
in Computers in Education Elsevier, IFIP p. 621-626, 1988.

[YOUNG & O'SHEA 81] R. M. Young, T. O'Shea
" Errors in children's subtraction "
Cognitive Science, 5, 153-77, 1981.

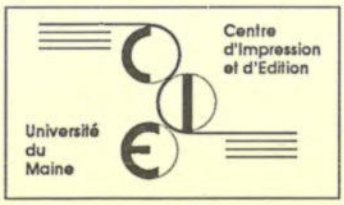
[YOUNGGREN 88] Geri Younggren
" Using an Object-Oriented Programming Language to Create Audience-Driven Hypermedia Environments "
in E. Barrett (ed) *" Text, ConText, and HyperText "*, p. 75-92, 1988.

[ZIMILES 77] H. Zimiles
" A radical and regressive solution to the problem of evaluation "
in L. G. Katz (ed.) *" Current topics in early childhood education "* Norwwod, Ablex Publishing Corporation, vol. 1, 1977.

[ZUIDENA & Al. 90] J. Zuidena, A. van Hienen, M. Streefkerk
" Interdisciplinarity in educational software design : an ITS for fractions "
Applica 90, Lille, 1990.

Bibliographie

Table des matières



Responsable de publication Martial Vivet, LIUM, Le Mans