# Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève

Dominique Py

# Quelques méthodes d'intelligence artificielle pour la modélisation de l'élève

**Dominique Py**

*IRISA*
*Campus Universitaire de Beaulieu*
*35042 Rennes cedex*

RÉSUMÉ : *Cet article présente quelques-unes des techniques d'intelligence artificielle qui ont été utilisées pour la modélisation de l'élève : la planification, l'apprentissage, le diagnostic et la révision de croyances. Chacune de ces techniques est décrite brièvement, puis son application à la modélisation de l'élève est présentée et illustrée par un ou deux exemples représentatifs. Les avantages et les inconvénients de ces différentes méthodes sont discutés. Enfin, nous dressons un bilan de ce tour d'horizon et évoquons quelques perspectives.*

ABSTRACT : *This paper presents some of the artificial intelligence techniques which have been used for student modeling : planning, machine learning, diagnosis and belief revision. Each technique is briefly described, then its application to student modeling is presented and illustrated by a few examples. The advantages and drawbacks of these methods are discussed. We end this survey with some directions for further researches.*

MOTS-CLÉS : *Tuteurs Intelligents, modélisation de l'élève, intelligence artificielle.*

KEYWORDS : *Intelligent Tutoring Systems, student modeling, artificial intelligence.*

## 1. Introduction

Les systèmes d'Enseignement Intelligemment Assisté par Ordinateur (ou EIAO), également appelés Environnements Interactifs d'Apprentissage avec Ordinateur, visent à faire acquérir des savoirs à un apprenant. Pour faciliter cette acquisition, il est souhaitable d'adapter l'interaction tuteur/élève en fonction de chaque élève. Cette personnalisation est obtenue par l'élaboration et l'exploitation d'un "modèle de l'élève", ensemble d'informations propres à un

apprenant. Ces informations peuvent être de natures diverses, mais le plus souvent elles portent sur les connaissances et les savoir-faire que le système attribue à l'élève au vu de son comportement. Sous le terme "modélisation de l'élève" sont regroupés des travaux divers, portant sur la modélisation cognitive des processus d'apprentissage, la représentation informatique de ces modèles, ou le diagnostic de l'état cognitif d'un élève. Dans cet article, nous nous intéressons principalement aux techniques d'intelligence artificielle qui permettent d'élaborer automatiquement un modèle de l'élève en cours d'interaction. Une grande variété de méthodes a été proposée depuis l'apparition de cette problématique. Dans un premier temps, elles étaient plutôt spécifiques au tuteur considéré, puis on a cherché à transposer des techniques provenant d'autres domaines de l'intelligence artificielle, qu'elles soient numériques ou symboliques.

L'objectif de cet article est de présenter quelques-unes des méthodes d'intelligence artificielle qui ont été appliquées à la modélisation de l'élève. Chacune d'elles est décrite brièvement, puis son application à la modélisation de l'élève est présentée et illustrée par un ou deux exemples représentatifs. Il ne s'agit pas de dresser une liste exhaustive des travaux dans le domaine, mais de décrire quelques réalisations intéressantes en essayant de dégager leurs avantages et leurs limites.

Le plan adopté dans cet article est le suivant : la seconde section est consacrée aux méthodes classiques de modélisation ; les sections trois à six décrivent quatre familles de travaux portant respectivement sur la planification, l'apprentissage, le diagnostic et la révision ; enfin la dernière section tire les conclusions de ce panorama.

## 2. Méthodes classiques de modélisation de l'élève

Les recherches sur les premiers tuteurs intelligents se sont focalisées sur la modélisation de l'expertise du domaine. Puis, on a ressenti la nécessité de disposer de connaissances sur l'élève au sein des tuteurs. Pour adapter l'interaction et la pédagogie à l'élève, il est apparu indispensable de disposer d'informations sur les aptitudes, les connaissances et les lacunes propres à chaque élève. Or, ceci était justement facilité par l'existence d'un corpus de connaissances constituées sur le domaine : le modèle de l'expert. Aussi est-il apparu naturel de prendre comme modèle de l'élève une copie, plus ou moins complète, du modèle de l'expert.

Dans un modèle de type "expertise partielle" (*overlay*), l'expertise est découpée en unités de base, et le modèle de l'élève se compose d'un sous-ensemble de ces entités. Aux deux extrêmes, un modèle vide correspond à l'élève qui n'aurait aucune connaissance du domaine, tandis qu'un modèle identique à celui de l'expert correspond à l'élève qui aurait atteint le même niveau de maîtrise qu'un expert du domaine. Chaque item de connaissance du modèle de l'élève peut être étiqueté par une valeur discrète (par exemple, connu/inconnu) ou continue (sur un intervalle allant de zéro à un).

Ce principe de modélisation a été conçu dans le cadre des tuteurs West [BB 82] et Wusor [GOL 82]. Il a été repris et étendu de manière à traiter explicitement les conceptions incorrectes dans les systèmes Buggy et Debuggy [BUR 82].

### 2.1. *West*

West est un tuteur basé sur un jeu à deux joueurs, de type "jeu de l'oie". L'élève joue contre l'ordinateur. West vise à faire pratiquer l'arithmétique à l'élève. Chaque joueur à son tour reçoit trois nombres tirés au hasard, avec lesquels il doit composer une expression arithmétique utilisant l'addition, la soustraction ou la multiplication. La valeur de l'expression obtenue indique le nombre de cases dont le joueur fait avancer son pion.

Les connaissances nécessaires à ce jeu sont d'une part des connaissances arithmétiques (bon emploi des opérateurs et des parenthèses), d'autre part des connaissances stratégiques sur le jeu (utilisation des raccourcis, des cases qui permettent de rejouer, etc.). Ces connaissances sont regroupées dans le modèle de l'expert, qui est ainsi capable de jouer de manière optimale. Après chaque coup joué par l'élève, le système détermine les connaissances effectivement utilisées par l'élève et les compare aux connaissances qu'aurait utilisées l'expert dans la même situation. Le modèle de l'élève mémorise, pour chaque connaissance, le nombre de fois où l'élève l'a utilisée à bon escient, le nombre de fois où l'élève l'a utilisée à mauvais escient, et le nombre de fois où l'élève ne l'a pas utilisée alors que l'expert l'a utilisée.

Le modèle de l'élève est ensuite consulté par le tuteur pour individualiser le guidage et les aides. Par exemple, le tuteur ne donne des conseils à l'élève que sur les connaissances que ce dernier maîtrise mal.

### 2.2. *Wusor*

Wusor est un tuteur basé sur un jeu qui fait manipuler des probabilités à l'élève. Les connaissances exprimées dans le modèle de l'expert sont de nature opératoire. Mais à la différence de West, où les connaissances ne sont pas structurées à l'intérieur du modèle, les connaissances dans Wusor sont organisées en un *graphe génétique*. Chaque nœud du graphe est une procédure élémentaire. Les arcs entre deux nœuds expriment les relations entre les procédures: généralisation/spécialisation, analogie, déviation/correction et simplification/affinement. Des connaissances proches sont donc représentées par des nœuds voisins dans le graphe génétique, en particulier les procédures visant le même but forment un îlot de nœuds interconnectés. Cette structuration du modèle permet de guider le diagnostic et l'enseignement. Le modèle de l'élève est constitué du sous-graphe représentant les connaissances attribuées à l'élève. Goldstein fait l'hypothèse que les connaissances nouvellement acquises sont proches des connaissances déjà acquises, et focalise le diagnostic à la frontière du modèle de l'élève. Si l'élève améliore ses performances, le tuteur attribuera

de préférence ce gain à l'acquisition de connaissances proches des connaissances déjà acquises.

Les interventions pédagogiques de Wusor, comme celles de West, sont guidées par le modèle de l'élève. Les conseils portent sur les connaissances proches de celles déjà acquises, qui sont supposées plus facilement assimilables par l'élève que des connaissances éloignées.

Dans les tuteurs tels que West ou Wusor, les erreurs commises par l'élève s'expliquent en termes d'absence de connaissances: c'est l'ignorance de telle règle ou de tel concept qui amène l'élève à ne pas jouer le meilleur coup possible. Une autre approche consiste à représenter dans le modèle de l'élève des règles dont l'application produit un résultat incorrect. Cette méthode a été utilisée par Burton et Brown dans les systèmes Buggy et Debuggy.

### 2.3. *Buggy et Debuggy*

En étudiant les comportements des élèves sur des problèmes d'arithmétique, Burton et Brown ont fait l'hypothèse que les erreurs proviennent de procédures élémentaires incorrectes plutôt que de difficultés à appliquer l'algorithme. Pour simuler le comportement d'un élève qui effectue des soustractions, ils ont réalisé le système Buggy, dans lequel les connaissances sont représentées par un réseau de procédures élémentaires. Toute procédure correcte peut être remplacée par une procédure incorrecte ayant le même domaine d'application (par exemple, n-0 = n devient 0-n = n). Un tel réseau de procédures correctes et incorrectes constitue un modèle de l'élève exécutable, qui effectue des soustractions de manière incorrecte mais cohérente, en commettant toujours les mêmes types d'erreurs. Ces principes de représentation ont été appliqués à la construction automatique d'un modèle de l'élève: à partir d'un ensemble de réponses données par un élève, Debuggy construit le réseau de procédures correctes et incorrectes dont le comportement se rapproche le plus de celui de l'élève.

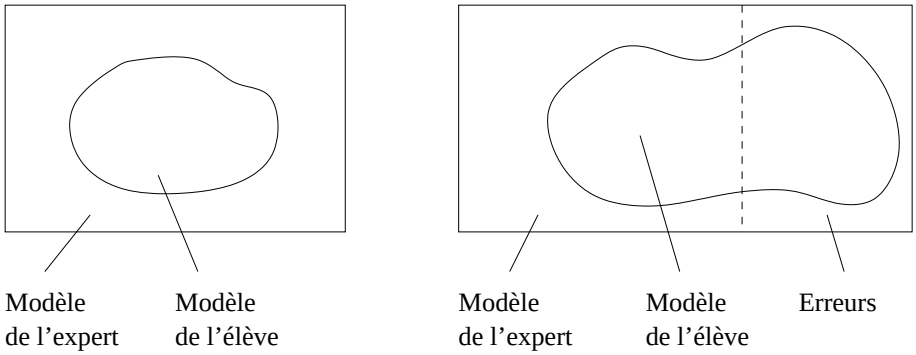


**Figure 2.1** : *Méthodes classiques de modélisation*

Les hypothèses de modélisation sous-jacentes à ces systèmes (considérer le

modèle de l'élève comme un sous-ensemble du modèle de l'expert, augmenté d'un ensemble de conceptions incorrectes) étaient à l'origine liées à des applications (West, Wusor, Buggy). Elles ont été reprises largement et ont donné lieu à de nombreuses réalisations, sans doute en raison de leur relative simplicité de mise en œuvre (réutiliser le modèle de l'expert, y adjoindre éventuellement des erreurs, ce qui nécessite cependant un recensement préalable).

### 2.4. *Formalisation logique du diagnostic*

Plus récemment, une formalisation logique de ces méthodes de diagnostic a été proposée par Aiello, Cialdea et Nardi [ACN 91]. Le diagnostic est vu comme un métaraisonnement, qui suppose de savoir résoudre des problèmes dans le domaine considéré, comparer la solution correcte à celle de l'élève et engendrer des hypothèses sur les conceptions incorrectes qui expliquent les réponses de l'élève. Pour cela, trois types de connaissances sont nécessaires: le modèle de l'expert E, le modèle de l'élève S et le catalogue des erreurs B. Ces structures constituent le niveau objet : chacune est une théorie du premier ordre, constituée d'une collection d'axiomes et d'un système logique pour effectuer des déductions. La formule-clé du métaniveau, reliant une théorie objet C à la métathéorie MT est la formule $derivable_C(x, y)$ où x et y sont des variables libres, telle que pour tout ensemble fini de formules $\Sigma$ et toute formule $\alpha$, s'il existe une preuve non triviale de $\alpha$ dans C utilisant exactement les formules de $\Sigma$ comme axiomes, alors :

$$MT \vdash derivable_C(\Sigma, \alpha)$$

La réponse de l'élève est la formule $\alpha$, les connaissances utilisées pour produire cette réponse forment l'ensemble $\Sigma$, la théorie objet C est E seul (pour une réponse correcte) ou bien $E \cup B$ (pour une réponse incorrecte). Expliquer la réponse de l'élève se ramène à construire une ou plusieurs dérivations, les comparer et choisir la meilleure suivant le critère souhaité.

Ce cadre logique peut être vu comme une formalisation abstraite des diagnostiqueurs implémentés dans des systèmes tels que Debuggy ou LMS [SLE 82].

## 3. Planification et reconnaissance de plan

Le raisonnement sur les plans est un thème important en intelligence artificielle. De manière classique, un plan est considéré comme une séquence ordonnée d'actions qui vise à satisfaire un but donné. Chaque action est définie par ses préconditions (les faits qui doivent être vrais pour que l'action soit exécutable) et ses effets (les modifications apportées à l'état du monde par l'exécution de l'action).

Les premiers travaux en planification portaient sur des actions et des plans déterministes, dans un monde dont l'état était supposé entièrement connu. Les recherches actuelles visent à traiter l'incertitude, l'imprécision et l'incomplétude qui caractérisent les situations réelles.

### 3.1. *Planification en EIAO*

Deux familles de travaux se distinguent. La *génération de plan* consiste à produire le plan qui permet de satisfaire un but donné (en EIAO, elle s'applique par exemple à la génération de plans pédagogiques [LFV 92]). La *reconnaissance de plan* est le processus réciproque : elle consiste à découvrir, à partir de l'observation de quelques actions, le plan mis en œuvre par le sujet observé. Elle s'applique à la modélisation de l'élève lorsque la résolution du problème ne peut s'obtenir par l'exécution pas à pas d'un algorithme (comme en arithmétique), mais requiert le choix d'actions adaptées, à l'aide d'heuristiques, et peut amener à des retours en arrière (comme en algèbre ou en géométrie). Pour ce type de problèmes, il existe en général plusieurs solutions d'efficacité variable. On considère alors ces différentes solutions comme des plans et le système cherche à reconnaître le plan exécuté par l'élève à partir des actions observées. Ceci permet de prévoir les prochaines actions, ou bien d'inférer les étapes intermédiaires non observées. La reconnaissance de plan suppose de connaître les plans existants : c'est une contrainte forte dans la mesure où leur nombre est généralement élevé. Pour contourner cet obstacle, on peut utiliser une bibliothèque de plans abstraits, qui seront spécialisés, ou bien générer ces plans à la demande.

La reconnaissance de plan permet d'élaborer un modèle local à une session, ou à un problème, puisqu'elle vise à inférer la stratégie mise en œuvre pour atteindre un but donné. En revanche, elle ne permet pas à elle seule de produire un modèle à long terme des connaissances et des savoir-faire de l'élève.

### 3.2. *Macsyma Advisor*

Un des premiers systèmes incluant une modélisation de l'élève basée sur la reconnaissance de plan est Macsyma Advisor [GEN 82]. Macsyma est un système de calcul formel, qui permet à l'utilisateur de manipuler et transformer des expressions symboliques. Un problème, dans Macsyma, consiste par exemple à résoudre une équation du second degré en lui appliquant des opérateurs tels que "développer" ou "factoriser".

Un problème peut être résolu par différentes séquences de commandes. Ces solutions sont regroupées dans un "plan" abstrait, arborescence exprimant les différents choix possibles et dont les actions comportent des variables. La reconnaissance de plan part de la séquence d'actions produite par l'élève, et cherche à lui faire correspondre l'un des plans possibles, en instanciant les variables du plan avec les valeurs des formules de l'élève. Il peut y avoir des écarts si l'élève a commis une erreur ou employé une action inadéquate. Dans ce cas, le système choisit le plan le plus proche de celui de l'élève, selon un critère de parcimonie : il considère que l'élève a fait le moins d'erreurs possible.

Une caractéristique importante de Macsyma Advisor est de représenter, associés à chaque action, les buts et les croyances qui lui sont liés. Ceci permet au conseiller de suggérer l'action appropriée aux objectifs de l'élève, lorsqu'une erreur est détectée.

### 3.3. *PHI, Mentoniezh*

Dans Macsyma Advisor, l'analyse du plan ne débute que lorsque l'utilisateur a réalisé entièrement sa séquence de commandes. Un autre contexte d'utilisation de la reconnaissance de plan est celui dans lequel le travail de l'élève est suivi étape par étape, en affinant le plan reconnu au fur et à mesure. C'est le cas de PHI [BBD 91], un système d'aide intelligent pour les utilisateurs d'Unix. Un générateur de plan produit, pour un but donné, des plans abstraits sous forme de formules logiques comportant des opérateurs modaux. Ces plans sont vus comme des candidats à l'explication des actions de l'utilisateur. Après chaque action ne sont conservés que les candidats cohérents avec les observations. De plus, les plans abstraits sont instanciés avec les commandes et paramètres des actions effectives de l'utilisateur. Enfin, s'il est nécessaire de retenir un plan unique parmi les candidats valides, un module de raisonnement probabiliste sélectionne un "meilleur plan" sur des critères de plausibilité.

Le tuteur Mentoniezh [PY 96], dans le domaine de la démonstration en géométrie élémentaire, est basé sur le même principe. Un démonstrateur automatique produit, pour un problème donné, un ensemble de démonstrations qui constituent les plans de résolution. Chaque étape de démonstration est une action du plan. Après chaque pas de preuve effectué par l'élève, on détermine l'ensemble des plans qui expliquent au mieux les actions observées.

Suivre le plan de l'utilisateur étape par étape permet de le conseiller plus efficacement. Par exemple, l'aide peut se focaliser sur la prochaine étape logique du plan reconnu.

## 4. Apprentissage symbolique

L'apprentissage symbolique automatique est un des thèmes d'intelligence artificielle qui a donné lieu au plus grand nombre d'applications en modélisation de l'élève. L'apprentissage automatique consiste à construire ou modifier la représentation d'un objet ou d'un concept. Cet apprentissage peut se faire par instruction (dialogue avec un expert humain), par l'exemple (observation et généralisation à partir d'instances positives et négatives du concept à apprendre) ou par découverte (exploration d'un domaine).

### 4.1. *Apprentissage à partir d'exemples*

La voie principalement explorée pour la modélisation de l'apprenant est l'apprentissage par l'exemple. On considère un ensemble de réponses données par l'élève (les exemples) et on cherche à en inférer les règles abstraites modélisant son comportement. Le modèle de l'élève ainsi obtenu est exécutable, comme celui construit par Debuggy, mais il s'en distingue par le fait que les règles qui le composent ne sont pas prédéfinies: elles sont construites par apprentissage à partir de primitives (des "briques" de base). Ces primitives sont neutres du point de vue de la correction. Il n'y a donc pas de notion de règle *a*

*priori* correcte ou incorrecte, c'est la combinaison de ces règles sur un problème donné qui produit un résultat correct ou erroné.

Si l'utilisation de primitives ouvre un champ de possibilités beaucoup plus vaste que les bibliothèques de règles correctes et incorrectes, en contrepartie, elle peut conduire à la synthèse de règles dont la sémantique n'est pas claire. C'est pourquoi il est souvent utile d'appliquer des filtres, basés sur des informations propres au domaine, afin d'écarter la construction de règles irréalistes.

### 4.2. *ACM*

Le système ACM (*Automated Cognitive Modeling* [OL 88]) est l'un des premiers systèmes ayant utilisé l'apprentissage automatique pour la modélisation de l'élève, sur le domaine de la soustraction. Un problème de soustraction est vu comme un ensemble de chiffres organisés en colonnes et en rangées (haut, milieu, résultat). Langley et Ohlsson font l'hypothèse que les procédures de l'élève peuvent être représentées par des règles de production. Ils définissent un ensemble d'opérations primitives pour la soustraction, telles que :

**AjouterDix(N,C,R)** : ajoute 10 au nombre N situé au rang R, colonne C

**Soustraire(N1,N2,C)** : calcule la différence entre les nombres N1 et N2 situés en colonne C, et écrit ce chiffre comme résultat pour la colonne C

**ChangerColonne(C)** : passe au traitement de la colonne située à gauche de la colonne courante

Ils définissent également un ensemble de conditions primitives telles que :

$$N \in Colonne_c, Rang_r$$
$$N_1 \geq N_2$$
$$Colonne_1 \text{ à droite de } Colonne_2$$
$$\ldots$$

ACM cherche à inférer les conditions des règles appliquées par l'élève qui effectue une soustraction. La résolution d'un problème de soustraction est vue comme un parcours dans un espace de recherche, dont les arcs sont des actions primitives et les nœuds des points de choix entre différentes actions possibles. Pour analyser la réponse d'un élève, ACM reconstruit le chemin solution, c'est-à-dire la séquence d'actions exécutée par l'élève. Ce chemin est obtenu par une exploration de type "meilleur d'abord", guidée par des heuristiques donnant la priorité au chemin le plus court et le plus proche du chemin solution de l'expert.

Dans une seconde phase intervient une technique d'apprentissage à partir d'exemples. Il s'agit, à partir du chemin solution, de déterminer les conditions d'application des opérateurs qui ont été utilisés par l'élève. Pour cela, ACM parcourt à nouveau le chemin solution, en déterminant à chaque point de choix tous les opérateurs applicables. L'opérateur effectivement appliqué par l'élève

constitue un exemple positif, les autres constituent les exemples négatifs. Un mécanisme de discrimination permet alors d'inférer les conditions d'application des opérateurs: ce sont les conditions qui écartent les exemples négatifs et retiennent l'exemple positif. On obtient des règles telles que:

$$\begin{gathered}\text{si } A \in Colonne_c, Rang_{r1} \\ \text{et } B \in Colonne_c, Rang_{r2} \\ \text{et } A \geq B \\ \text{alors } Colonne_c, Rang_{bas} := A - B\end{gathered}$$

Le résultat final est un ensemble de règles qui reproduit le comportement de l'élève.

Dans les versions les plus récentes d'ACM, recherche de chemin et construction des conditions sont menées de front. Les règles déjà inférées servent à guider la recherche, ce qui assure que le chemin suivi reste consistant avec le modèle partiel déjà élaboré. C'est seulement lorsque le chemin suivi n'est expliqué par aucune règle connue que ACM déclenche l'apprentissage d'une nouvelle règle, qui sera incorporée au modèle.

### 4.3. *Programmation Logique Inductive*

Les développements récents de la programmation logique inductive (PLI) ont donné lieu à des recherches originales en modélisation de l'élève. La PLI est un domaine de l'apprentissage automatique, dont le but est d'inférer de façon inductive une théorie logique, à partir de faits devant être prouvés par cette théorie. Le principe général s'énonce ainsi: étant donnés

- une théorie logique $B$, décrivant le domaine;
- un ensemble de faits $E^+$, qui doivent être prouvés par la théorie solution, les éléments de $E^+$ sont appelés exemples, ou exemples positifs;
- un ensemble de faits $E^-$, qui doivent ne pas être prouvés par la théorie solution, les éléments de $E^-$ sont nommés contre-exemples, ou exemples négatifs;

trouver une hypothèse $H$ (ensemble de clauses) telle que:

- $B \wedge H \models E^+$ (la théorie explique les exemples);
- $B \wedge H \not\models E^-$ (la théorie écarte les contre-exemples).

$B \wedge H$ constitue la théorie solution.

L'application à la modélisation de l'élève se fait avec les correspondances suivantes:

- $B$ est l'ensemble des primitives servant à décrire les procédures recherchées;
- $E^+$ est l'ensemble des couples (problème, réponse) provenant de l'interaction avec l'élève;

- $E^-$ est vide : en effet, les observations faites sont toutes des exemples du comportement du sujet, des contre-exemples ne sont donc pas disponibles par le biais de l'observation ;

- $H$ est une représentation du raisonnement mis en œuvre par l'élève.

Emmanuel Siou [SIO 94] applique la PLI à la reconstruction du raisonnement d'un sujet (souffrant de troubles du langage) sur des exercices de transcodage (passage d'un nombre en lettres à un nombre en chiffres et réciproquement). Il utilise un ensemble de primitives décrivant des manipulations élémentaires sur les nombres et reconstruit automatiquement le raisonnement du patient sous forme d'un enchaînement de plusieurs de ces primitives exprimées par des clauses Prolog. Par exemple, si le nombre à transcoder est "vingt mille vingt" et la réponse du patient "20100020", on obtient :

| Procédure induite | Exécution sur l'exemple |
|---|---|
| transcode([],[]).<br>transcode([A\|B],N2)←<br>traduit(A,C),<br>segmente(C,D),<br>transcode(B,E),<br>concat(D,E,N2). | transcode([vingt,mille,vingt],[*2,0,1,0,0,0,2,0*])←<br>traduit([vingt],*20*),<br>segmente(20,*[2,0]*),<br>transcode([mille,vingt],[*1,0,0,0,2,0*]),<br>concat([2,0],[1,0,0,0,2,0],[*2,0,1,0,0,0,2,0*]). |

Autrement dit, la procédure suivie par le sujet consiste à traduire séparément chaque mot en un chiffre, puis à concaténer les trois chiffres obtenus.

La modélisation de l'utilisateur par PLI permet donc l'inférence de procédures complexes à partir de primitives et de couples question-réponse. De plus, il est possible d'affiner le diagnostic en choisissant judicieusement les questions à soumettre : lorsque des hypothèses concurrentes se présentent, le système génère des exemples, par le biais d'exercices, dans le but de départager ces hypothèses. Une limitation de cette méthode est qu'elle ne prend pas en compte les contradictions ni l'évolution du sujet : tout raisonnement inféré est ajouté au modèle de l'élève, et l'on risque d'obtenir à terme un modèle extrêmement général qui ne serait plus représentatif de l'état cognitif du sujet. Surmonter cette difficulté nécessiterait l'introduction d'un mécanisme qui "oublie" les raisonnements qui ne sont plus utilisés, ou bien qui les hiérarchise en fonction de leur emploi effectif.

## 5. Diagnostic basé sur un modèle

Le diagnostic en intelligence artificielle (diagnostic de pannes, diagnostic médical) est abordé sous deux points de vue différents. L'approche "système-expert" est la plus ancienne. Elle consiste à définir des règles exprimant les relations qui existent entre les symptômes (de panne, de maladie) et les diagnostics. Un exemple représentatif de cette approche est le système MYCIN, sur

le domaine des maladies infectieuses, qui produit un diagnostic médical à partir d'un ensemble de symptômes et de résultats partiels d'analyses. L'approche basée sur les modèles (*model-based*) est plus récente. Elle nécessite de disposer d'un modèle de fonctionnement du dispositif considéré. Ce modèle est utilisé pour prédire le comportement attendu du système. La comparaison entre les observations et les prédictions permet de déterminer, de manière logique, l'état des composants de manière à identifier la source de la panne. L'approche basée sur les modèles présente l'avantage d'être indépendante du dispositif considéré.

### 5.1. *Exemple*

Pour illustrer cette approche, considérons l'exemple classique d'un dispositif de calcul arithmétique composé de deux additionneurs (A1 et A2) et trois multiplicateurs (M1, M2 et M3). Le fonctionnement de ces composants est décrit de manière logique par des formules telles que :

$Additionneur(Ad) \wedge \neg EnPanne(Ad) \wedge Entre1(Ad, u) \wedge Entre2(Ad, v) \wedge Somme(u, v, w) \rightarrow Sortie(Ad, w)$
qui se lit : "Si Ad est un additionneur, s'il n'est pas en panne, s'il reçoit en entrée les valeurs u et v dont la somme est w, alors Ad fournit la valeur w en sortie"

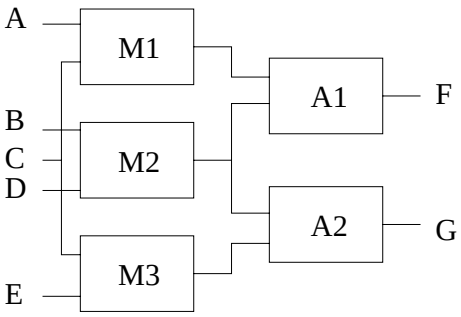


**Figure 5.1** : *Circuit arithmétique*

Si l'on donne en entrée les valeurs A=3, B=2, C=2, D=3, E=3 et que l'on observe en sortie les valeurs G=12, F=10 (la valeur correcte pour F est 12), plusieurs diagnostics minimaux se présentent : soit A1 est en panne, soit M1 est en panne, soit M2 et A2 sont en panne, soit M2 et M3 sont en panne. Ces diagnostics sont minimaux au sens où il suffit que A1 soit en panne pour expliquer l'observation, mais d'autres composants peuvent être également en panne (on considère qu'un composant en panne fournit un résultat aléatoire). D'autres critères peuvent intervenir pour sélectionner un diagnostic parmi les diagnostics minimaux, par exemple on peut préférer les diagnostics impliquant le plus petit nombre de composants (ici, les diagnostics {A1} et {M1}).

### 5.2. *Diagnostic cognitif basé sur un modèle*

Les méthodes classiques de modélisation de l'élève relèvent plutôt de l'approche "système-expert" : des règles associent des classes de réponses à des conceptions sous-jacentes. Peut-on appliquer l'approche basée sur les modèles à la modélisation de l'élève? John Self a étudié la question en illustrant son propos sur le domaine de la soustraction [SEL 92]. Il définit un circuit effectuant une soustraction à trois colonnes, de la forme ABC − DEF = GHI, à partir de plusieurs types de composants: un "comparateur", un "soustracteur", un "calculateur de retenue". Il montre que la méthode de diagnostic, appliquée à ce modèle, permet de déterminer automatiquement la source de l'erreur de l'élève. Par exemple, si on soumet le problème "274 − 129" et la réponse incorrecte "155" (au lieu de 145), les diagnostics minimaux obtenus indiquent que l'erreur provient soit du calcul de la retenue dans la première colonne, soit du report de cette retenue dans la seconde colonne, soit de la soustraction effectuée dans la seconde colonne.

En diagnostic de circuits, les composants en panne sont considérés comme ayant un comportement aléatoire, c'est-à-dire que le résultat produit est totalement imprévisible. Or, en modélisation de l'élève, certaines erreurs sont plus fréquentes que d'autres. Self montre qu'il est possible d'inclure ces erreurs spécifiques dans la description des composants, et que ceci réduit le nombre de diagnostics candidats. Il remarque également que le diagnostic devrait être basé sur plusieurs observations, dans le cadre d'une interaction avec un élève.

Dans sa conclusion, Self soulève plusieurs problèmes non résolus: le fait que l'élève évolue et peut "s'auto-réparer", l'accent mis implicitement sur les erreurs de l'élève, le fait qu'il n'existe pas de "circuit" prédéfini qui décrive les connaissances. Si l'application du diagnostic basé sur les modèles à la modélisation de l'élève est une idée séduisante, elle risque en effet de se heurter à la difficulté de concevoir un circuit pour les domaines complexes, lorsqu'il n'existe pas d'algorithme de résolution. De plus, ce que l'on cherche à capturer est plutôt l'architecture du circuit que l'état des composants. L'élève peut commettre une erreur due au fait que l'architecture de son "circuit" est différente de l'architecture présumée, même si aucun de ses "composants" n'est en panne: le diagnostic produit n'est alors pas valide. Par exemple l'élève peut maîtriser les opérations de base telles que la comparaison ou la soustraction de deux chiffres, mais ne pas avoir compris le principe de la retenue.

En conclusion, la proposition de Self va dans le sens d'une formalisation plus poussée de la modélisation de l'élève, mais elle n'est envisageable dans la pratique que pour des domaines simples.

## 6. Incertitude et évolution dans le temps

La prise en compte des aspects dynamiques du modèle de l'élève présente plusieurs difficultés. D'une part, le contenu du modèle est inféré à partir du comportement de l'élève, donc les informations qu'il contient ne sont pas cer-

taines. D'autre part, l'élève évolue au cours de l'interaction. Enfin, l'élève peut donner des réponses contradictoires.

Considérons l'exemple suivant. L'élève donne une réponse R, que l'on explique en supposant qu'il connaît le fait A. La formule *sait(A)* est ajoutée au modèle. Une règle d'inférence exprime que, si l'élève connaît A, il est très probable qu'il connaisse également B. La formule *sait(B)* est donc ajoutée au modèle. Puis, l'élève donne une réponse R', que l'on explique en supposant qu'il ne connaît pas le fait B. La formule *non(sait(B))* est donc ajoutée au modèle. A ce stade, le modèle de l'élève est contradictoire, puisqu'il contient à la fois *sait(B)* et *non(sait(B))*. Plusieurs explications peuvent être apportées à cela: la règle d'inférence est trop forte et l'élève connaît A sans connaître B, ou bien l'analyse de la réponse R' ne permettait pas de conclure que l'élève ne connaît pas B, ou encore l'élève a "oublié" B entre les deux réponses. Le choix de retirer l'une des deux formules pour restaurer la cohérence va se faire sur des critères qui permettent de choisir entre les explications possibles: on pourra choisir de privilégier l'information la plus récente, ou la plus directement obtenue, ou encore la plus sûre (selon des critères donnés).

Le traitement des contradictions et de l'évolution des connaissances de l'élève n'a été abordé que depuis peu, et par un nombre restreint de travaux [BAR 88]. Ce problème est particulièrement délicat car il s'ajoute à celui de l'inférence du modèle. C'est pourquoi la plupart des travaux font l'hypothèse que les connaissances de l'élève sont statiques et se focalisent sur l'inférence de l'état cognitif, supposé cohérent, à partir du comportement.

En intelligence artificielle, des problèmes similaires sont rencontrés dans la mise à jour de bases de données ou de théories logiques. Lorsqu'arrive une information nouvelle, contradictoire avec le contenu de la base, on cherche à incorporer cette information en modifiant la base de manière à éliminer la contradiction. Certains travaux abordent ce thème sous l'angle de la "révision de théorie". D'autres se basent sur les systèmes consacrés au maintien de la cohérence, les TMS (*Truth Maintenance Systems*), définis par Doyle [DOY 79]. Un TMS gère un ensemble d'assertions (qui évolue) et leurs conséquences logiques. Lorsque l'arrivée d'une nouvelle assertion amène une contradiction, le système recherche une assertion dont la suppression permet de rétablir la cohérence, et la retire de l'ensemble. Ces deux approches, révision de théorie et TMS, ont inspiré des travaux en modélisation de l'élève.

### 6.1. *Maintien de la cohérence*

Huang, McCalla et Greer définissent un système de maintenance du modèle de l'élève qui manipule des connaissances par défaut et permet la révision des connaissances, en combinant des techniques issues des TMS et du diagnostic [HCG 90]. Leur modèle se veut général et il est appliqué, pour les besoins de l'exemple, à un tutoriel du langage Lisp. Les auteurs se placent délibérément en aval du processus d'analyse des réponses de l'élève: leur système travaille directement avec les connaissances attribuées à l'élève.

Le modèle de l'élève contient deux ensembles de connaissances :

- des "connaissances confirmées", qui ont été obtenues par l'analyse des réponses de l'élève ;
- des connaissances par défaut, informations sur les connaissances usuelles des élèves, qui sont utilisées lorsqu'aucune information plus précise n'est disponible.

Le premier est appelé CBK (*Confirmed Body Knowledge*) et le second DBK (*Default Body Knowledge*). Chacun de ces ensembles peut être soumis à la révision.

**Révision du CBK.** Le CBK contient deux types d'éléments : les prémisses (connaissances directement extraites des réponses de l'élève, ou bien règles d'inférence de connaissances) et les connaissances inférées (obtenues par ces règles). Cet ensemble croît lorsque de nouvelles connaissances sont inférées grâce aux règles ou bien obtenues par l'analyse d'une réponse de l'élève. L'ajout de nouvelles informations peut mener à une contradiction : dans ce cas, le système doit retirer un ensemble minimal de prémisses de manière à rétablir la cohérence. Lorsque plusieurs ensembles minimaux existent, il est nécessaire d'utiliser des connaissances propres au domaine pour sélectionner le plus plausible.

**Révision du DBK.** Le DBK est exprimé sous la forme d'un graphe acyclique orienté, dont les nœuds sont des ensembles de connaissances étiquetés par des valeurs (novice, moyen, expert ou inconnu) qui qualifient le degré de maîtrise qu'a l'élève des concepts. La valeur d'un nœud est contrainte par celle de ses voisins. Par exemple, si l'élève connaît mal les fonctions récursives (le concept "fonction récursive" possède la valeur "novice"), il ne peut maîtriser les fonctions en général (le concept "fonction" ne doit pas posséder la valeur "expert"). La mise à jour du DKB se fait par propagation de valeurs. La révision du CKB peut entraîner la mise à jour d'une ou plusieurs valeurs du DKB, qui entraîne par ricochet la mise à jour des valeurs des nœuds voisins, jusqu'à ce que les contraintes soient satisfaites.

Ce système constitue la première application des techniques de TMS à la modélisation de l'élève et aborde de front l'évolution du modèle de l'élève. La première partie (révision du CBK) propose une solution robuste et élégante au traitement de la non-monotonie. La seconde partie (révision du DKB) est moins convaincante dans la mesure où la sémantique des changements effectués n'apparaît pas clairement. En conclusion, les auteurs notent qu'il serait intéressant d'étendre leur modèle de manière à pouvoir exprimer une relative incohérence dans les croyances de l'élève.

### 6.2. *Révision de théorie*

Dans un autre contexte, Paiva, Self et Hartley proposent un système de maintenance du modèle de l'élève assez proche de la révision de théorie [PSH 94]. Ils considèrent deux types d'évolution, qui sont traités de manière différente : l'évolution du système, due à l'incertitude du processus d'acquisition, et l'évolution de l'élève.

Le modèle de l'élève est représenté par une base de faits, exprimés dans un langage logique. Trois types d'évolution du système, classiques en révision de théorie, sont décrits :

**Expansion :** Ajout d'un fait, qui ne perturbe pas la cohérence

**Mise à jour :** Modification d'un fait, qui ne perturbe pas la cohérence

**Révision :** Modification de la base lorsqu'une incohérence est détectée

Le point délicat est la révision. Il s'agit de rétablir la cohérence tout en minimisant les changements apportés à la base. Le choix des faits à retirer est basé sur un critère de rationalité, qui dépend de la "confiance" que le système accorde aux faits.

De même, trois types d'évolution de l'élève sont décrits :

**Evolution élémentaire :** Expansion (ajout d'un fait), révision (modification d'un fait) et contraction (retrait d'un fait)

**Evolution de la structure :** Découpage (découpe un ensemble de faits en deux ou plus, de manière à conserver la cohérence, ce qui permet de faire cohabiter deux points de vue opposés) et regroupement (regroupe deux ensembles de faits compatibles en un seul)

**Evolution de l'usage :** Activation (fait passer le statut d'un fait de "passif" à "actif") et acceptation (fait passer le statut d'un fait de "actif" à "passif"), qui expriment la séparation entre mémoire à court terme et mémoire à long terme.

Cette approche présente l'avantage de ne pas être liée à un domaine particulier. Le système de maintenance est indépendant du module d'acquisition et d'analyse des réponses de l'élève: il gère un ensemble de faits, exprimés dans un formalisme logique, à partir des informations qu'il reçoit du module d'acquisition. Si cette méthode est bien adaptée à des connaissances factuelles, en revanche, l'extension aux connaissances procédurales n'est pas immédiate et nécessiterait des aménagements importants.

## 7. Conclusion

Les techniques d'intelligence artificielle ont incontestablement donné lieu à une grande variété d'applications en modélisation de l'élève. Ceci peut s'expliquer par la richesse de cette problématique, dont chacune des facettes (anticipation du comportement, construction d'un modèle exécutable, diagnostic d'erreurs, évolution) peut être abordée dans un cadre théorique à l'aide d'outils adaptés.

La coopération entre intelligence artificielle et modélisation de l'élève se révèle fertile dans les deux sens : certains auteurs ont été amenés à proposer des variantes ou des améliorations des techniques utilisées, améliorations qui dépassent le cadre strict de la modélisation de l'élève. Par exemple, Siou a défini un algorithme de recherche bidirectionnelle pour la PLI, afin de surmonter les limitations rencontrées par les algorithmes unidirectionnels classiques, dans le cadre de la modélisation du patient.

Cependant, les différentes familles de travaux restent encore relativement séparées. Il n'existe pas d'algorithme général pour la modélisation de l'élève, même si la description formelle de Aiello, Cialdea et Nardi constitue une avancée dans ce sens. La diversité des disciplines enseignées et des types d'interaction est probablement un obstacle sérieux à l'émergence d'algorithmes transversaux, de même que la variété des options pédagogiques prises par les concepteurs.

Si les méthodes et les algorithmes ont été largement étudiés, en comparaison, moins d'attention a été portée jusqu'ici à la représentation des connaissances au sein du modèle de l'élève. Les représentations basées sur les faits et les règles prédominent. Or, les modèles psychologiques et cognitifs offrent des structures élaborées qui pourraient inspirer des représentations à la fois plus complexes et plus réalistes que celles employées actuellement [VER 91]. D'autre part, le recours excessif à des méthodes de modélisation sophistiquées risque de mettre l'accent sur le formalisme et les manipulations syntaxiques, au détriment de la signification des modèles obtenus. De manière générale, la recherche en modélisation de l'élève a tout intérêt à confronter davantage ses résultats avec ceux de la psychologie et de la didactique.

Enfin, si la validation informatique des modèles proposés est souvent réalisée par l'implémentation et les tests, il reste à effectuer la validation pédagogique de ces modèles dans de réelles situations d'enseignement. Cette validation soulève des problèmes non négligeables, en raison des difficultés pratiques de la mise en œuvre d'expérimentations en classe, mais aussi du fait que le modèle de l'élève n'est qu'un élément du tuteur. Sa validation suppose que le tuteur complet soit implémenté, ce qui n'est pas toujours le cas, et elle ne peut se faire indépendamment de la validation du tuteur lui-même : distinguer les effets dûs en propre au modèle de l'élève dans l'interaction est une tâche quasi impossible. Néanmoins, l'expérimentation de tuteurs intégrant des modèles "intelligents" de l'élève reste un objectif majeur et constitue une étape importante vers une utilisation effective de ces modèles dans les logiciels pour l'enseignement.

## Bibliographie


[ACN 91] Aiello L., Cialdea M., Nardi D., « A meta-level abstract description of diagnosis in intelligent tutoring systems », *Proceedings of the 6th international PEG conference*, Rapallo, Italie, 1991, p. 437-442.

[BAR 88] Baron M., « Quelques problèmes de non-monotonie en EIAO », *Actes du premier congrès européen Intelligence Artificielle et Formation*, Lille, 3-5 octobre 1988, p. 143-155.

[BB 82] BURTON R.R., BROWN J.S., « An investigation of computer coaching for informal learning activities », *Intelligent Tutoring Systems*, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 79-98.

[BBD 91] BAUER M., BIUNDO S., DENGLER D., KOEHLER J., PAUL G., « PHI – A Logic-Based Tool for Intelligent Help Systems », *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, IJCAI-93, Chambéry, France, 1993, p. 460-466.

[BRU 97] BRUILLARD E., *Les machines à enseigner*, Hermès, Paris, 1997.

[BUR 82] BURTON R.R., « Diagnosing bugs in a simple procedural skill », *Intelligent Tutoring Systems*, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 157-184.

[DOY 79] DOYLE E., « A truth maintenance system », *Artificial Intelligence*, vol. 12, n° 3, 1979, p. 231-272.

[GEN 82] GENESERETH M., « The role of plans in intelligent teaching systems », *Intelligent Tutoring Systems*, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 137-156.

[GMC 94] GREER J.E., MCCALLA I. (eds), *Student Modelling : the key to individualized knowledge-based instruction*, NATO ASI Series F vol. 125, Springer-Verlag, 1994.

[GOL 82] GOLDSTEIN I.P., « The genetic graph : a representation for the evaluation of procedural knowledge », *Intelligent Tutoring Systems*, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 51-78.

[HCG 90] HUANG X., MCCALLA I., GREER J.E., « Student Model Revision : Evolution and Revolution », *Proceedings of the 8th biennal conference of the Canadian Society for Computational Studies of Intelligence*, AI CSCSI-90, Université d'Ottawa, Ontario, Canada, 1990, p. 98-105.

[LFV 92] LABAT J.M., FUTTERSACK M., VIVET M., « Planification pédagogique : de l'expertise humaine à sa modélisation dans un STI », *Proceedings of the second international Intelligent Tutoring Systems conference*, ITS-92, C. Frasson, G. Gauthier et G.I. McCalla (eds), LNCS 608, Springer-Verlag, 1992, p. 515-522.

[OL 88] OHLSSON S., LANGLEY P., « Psychological evaluation of path hypotheses in cognitive diagnosis », *Learning Issues for Intelligent Tutoring Systems*, H.Mandl et A.Lesgold (eds), Springer-Verlag, 1988, p. 42-62.

[PSH 94] PAIVA A., SELF J., HARTLEY R., « On the dynamics of learner models », *Proceedings of the 11th European Conference on Artificial Intelligence*, ECAI-94, A. Cohn (éd.), John Wiley and Sons Ltd, 1994, p. 178-196.

[PY 96] PY D., « Aide à la démonstration en géométrie : le projet Mentoniezh », *Sciences et Techniques Educatives*, vol. 3, n° 2, Hermès, Paris, 1996, p. 227-256.

[SEL 92] SELF J., « Cognitive diagnosis for tutoring systems », *Proceedings of the 10th European Conference on Artificial Intelligence*, ECAI-92, B. Neumann (éd.), John Wiley and Sons Ltd, 1992, p. 699-703.

[SIO 94] SIOU E., Programmation logique inductive et modélisation de l'apprenant : application à l'analyse des erreurs de raisonnement chez l'aphasique, thèse de l'université de Rennes I, 1994.

[SLE 82] SLEEMAN D.H., « Assessing Competence in Basic Algebra », *Intelligent Tutoring Systems*, D.Sleeman et J.S.Brown (eds), Academic Press, New York, 1982, p. 185-200.

[VER 91] VERGNAUD G. (éd.), « Les sciences cognitives en débat », *Première école d'été du CNRS sur les sciences cognitives*, Editions du CNRS, Paris, 1991.

[WEN 87] WENGER E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publisher, Los Altos, California, 1987.

**Titre en anglais**

Artificial intelligence methods and student modeling

**Biographie**

Dominique Py est maître de conférences en informatique. Elle a soutenu sa thèse en 1990 à l'université de Rennes I, dans le cadre du projet *Mentoniezh*. Elle effectue actuellement sa recherche à l'Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA) de Rennes, dans l'équipe Aïda. Ses principaux centres d'intérêt sont la conception des environnements d'apprentissage et la modélisation de l'utilisateur.