Systèmes experts et traitement des connaissances : vers un nouveau bond en avant de l'informatique individuelle?

Autor(en): **Bonzon, Pierre**

Objekttyp: **Article**

Zeitschrift: **Revue économique et sociale : bulletin de la Société d'Etudes Economiques et Sociales**

Band (Jahr): **47 (1989)**

Heft 4

PDF erstellt am: **04.06.2024**

Persistenter Link: https://doi.org/10.5169/seals-139858

Nutzungsbedingungen

Die ETH-Bibliothek ist Anbieterin der digitalisierten Zeitschriften. Sie besitzt keine Urheberrechte an den Inhalten der Zeitschriften. Die Rechte liegen in der Regel bei den Herausgebern.
Die auf der Plattform e-periodica veröffentlichten Dokumente stehen für nicht-kommerzielle Zwecke in Lehre und Forschung sowie für die private Nutzung frei zur Verfügung. Einzelne Dateien oder Ausdrucke aus diesem Angebot können zusammen mit diesen Nutzungsbedingungen und den korrekten Herkunftsbezeichnungen weitergegeben werden.
Das Veröffentlichen von Bildern in Print- und Online-Publikationen ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Die systematische Speicherung von Teilen des elektronischen Angebots auf anderen Servern bedarf ebenfalls des schriftlichen Einverständnisses der Rechteinhaber.

Haftungsausschluss

Alle Angaben erfolgen ohne Gewähr für Vollständigkeit oder Richtigkeit. Es wird keine Haftung übernommen für Schäden durch die Verwendung von Informationen aus diesem Online-Angebot oder durch das Fehlen von Informationen. Dies gilt auch für Inhalte Dritter, die über dieses Angebot zugänglich sind.

Ein Dienst der *ETH-Bibliothek*
ETH Zürich, Rämistrasse 101, 8092 Zürich, Schweiz, www.library.ethz.ch

http://www.e-periodica.ch

Systèmes experts et traitement des connaissances: vers un nouveau bond en avant de l'informatique individuelle?

Pierre Bonzon
Professeur
Institut d'informatique et organisation
Ecole des HEC
Université de Lausanne

INTRODUCTION

Dans l'histoire du développement de l'informatique, les années 1980 resteront marquées par l'avènement de l'informatique individuelle. Avec le recul dont nous disposons aujourd'hui, il apparaît que cet essor est dû, pour une bonne part, à la conjugaison de deux facteurs:

1. l'existence d'un produit standard, à savoir l'ordinateur personnel *IBM PC* associé au système d'exploitation *MS-DOS*, imposant de facto une norme technique à tout constructeur de matériel et de logiciel désireux d'offrir un produit compétitif;
2. la disponibilité, sur ces machines, d'un certain nombre de logiciels d'application, tels que *tableurs, systèmes de traitement de texte* ou de *gestion de base de données.*

Pour ce qui concerne l'existence d'une norme technique de facto, son adoption par les fournisseurs de systèmes compatibles aura permis la création d'un vaste marché, favorisant les échanges de toute nature, aussi bien entre constructeurs qu'entre utilisateurs, et aboutissant, grâce à une pression sur les prix très vive, à une véritable démocratisation de l'informatique. Que l'on songe, pour s'en convaincre, au cas d'une institution comme l'Ecole des HEC: grâce aux ordinateurs personnels, le cercle des utilisateurs de l'informatique, restreint jusque-là aux professionnels avertis, s'étend désormais aux professeurs de toutes les orientations.

Quant aux logiciels d'application, vendus par millions de copies à des prix très attractifs, leurs fonctionnalités simples se sont avérées parfaitement adaptées aux besoins des utilisateurs. La combinaison de ces différentes fonctionnalités dans des logiciels dits *intégrés*, tels que Multiplan, Lotus 1 2 3 ou Framework, aura, dans ce domaine, apporté la touche finale de confort attendue de tous.

LOGICIELS INTEGRES ET TRAITEMENT DES DONNEES

Rappelons, si besoin est, le principe des logiciels intégrés: les applications y sont prises en charge par un ensemble de programmes reposant sur un langage de commande unique, facilitant le transfert des données d'une application à l'autre. Les services offerts par ces logiciels permettent, grosso modo, de réaliser facilement toute application pouvant se ramener à

a) saisir des données, qu'elles soient numériques, textuelles, ou graphiques;

b) stocker ces données, à les consulter et à les mettre à jour à tout moment;

c) utiliser ces données pour effectuer des traitements divers, tels que tri, sélection, calcul, et à en restituer les résultats sous des formes appropriées.

Toute application de ce type se distingue par le caractère algorithmique, *entièrement déterminé à l'avance*, du traitement à effectuer, ne laissant aucune prise au hasard ou à l'incertitude. Considérons, pour illustrer ce point, l'emploi d'un *tableur* pour établir les *budgets prévisionnels* d'une entreprise, correspondant à différents scénarios. Chacun de ces scénarios, représentant une manière d'imaginer l'avenir, peut être défini à l'aide d'un ensemble de paramètres, tels que taux de croissance, évolution des prix, part du marché, etc. Une fois ces paramètres choisis *par l'utilisateur*, le système exécutera une séquence de traitements bien définie, correspondant à des formules connues, enregistrées dans le tableau avec les données historiques sur les ventes et les coûts de production. Le processus de quantification de l'incertitude, nécessaire pour déterminer les paramètres d'un scénario, ne pourra pas, quant à lui, être effectué simplement. La démarche à suivre dans ce but pourra être diverse: selon les cas, elle impliquera le recours à des modèles économétriques et l'exploitation explicite de liens de causalité, ou s'appuyera sur la seule intuition du planificateur.

VERS UN FORMALISME NOUVEAU: LES SYSTEMES EXPERTS

Parallèlement à ce type d'application, se profile depuis peu une nouvelle catégorie de modèles de traitement, née de la recherche en intelligence artificielle[1], et connue sous le nom de *systèmes experts*. Pour reprendre l'exemple évoqué ci-dessus, l'utilisation de tels systèmes devrait permettre de générer *automatiquement* des scénarios, l'utilisateur se contentant, en dernier ressort, d'en apprécier la vraisemblance ou de les pondérer pour obtenir un budget prévisionnel plausible.

Pour ce faire, un système expert doit être en mesure de:

a) saisir des connaissances d'expert;

b) stocker ces connaissances, les consulter et les mettre à jour à tout moment;

c) utiliser ces connaissances pour obtenir des données marquantes.

Dans notre exemple, les connaissances requises pour quantifier l'incertitude, en explicitant au besoin des liens de causalité, devraient être enregistrées dans le système, au même

titre que les données numériques concernant les ventes et les coûts de production, ou que les formules définissant le résultat d'exploitation; de même, le processus de raisonnement, permettant dans notre exemple d'obtenir les paramètres d'un scénario à partir de ces connaissances, devrait-il être contenu dans le système, au même titre que le programme calculant le résultat d'exploitation. De tels systèmes intégrés existent: notons d'emblée que l'on appelle *base de connaissances* l'ensemble des données, en général symboliques, enregistrées dans un système expert et susceptibles d'être mises en oeuvre dans un raisonnement, et *moteur d'inférence*, tout programme capable d'effectuer automatiquement des raisonnements en s'appuyant sur une base de connaissances quelconque.

VERS UNE DIMENSION NOUVELLE: L'EXPLOITATION DES CONNAISSANCES

Dans la perspective de ce que nous développerons plus loin, deux points méritent d'être précisés ici:

— le concept de *connaissance*, par opposition à celui de *donnée* utilisé dans les systèmes informatiques traditionnels;

— les conditions d'utilisation des programmes de traitement des connaissances.

On dit parfois, de toute connaissance, qu'elle est réflexive, c'est-à-dire qu'elle s'applique à elle-même; ou encore, qu'elle a conscience d'elle-même. Or, «savoir que l'on sait» ne signifie-t-il pas, d'un point de vue pratique, «être en mesure d'utiliser ses connaissances lorsqu'elles sont requises»? Toute connaissance se présente ainsi comme un *ensemble de données représentées avec leurs conditions d'utilisation*, ou, en d'autres termes, *avec une définition de leur finalité.*

Exemple

Considérons la connaissance historique élémentaire associée à la phrase suivante: «la bataille de Marignan eut lieu en 1515».

Imaginons, dans un premier temps, son enregistrement dans une base de connaissances sous la forme suivante:

SI objet=bataille & nom= Marignan

ALORS date= 1515

De toute évidence, la finalité associée à cette représentation est de fournir une date, et rien d'autre. Pour fournir un nom de bataille, et pouvoir ainsi répondre à la question «quelle est la bataille qui eut lieu en 1515?», il faudrait disposer d'une autre représentation:

SI objet=bataille & date= 1515

ALORS nom= Marignan

Tout moteur d'inférence capable d'interpréter la forme commune, particulièrement simple, de ces deux représentations, serait en mesure d'exploiter la connaissance qu'elles renferment.

Songeons, dans un deuxième temps, à enregistrer la phrase donnée plus haut dans sa forme textuelle, à savoir «la bataille de Marignan eut lieu en 1515». Si cet unique enregistrement contiendra toute l'information nécessaire pour répondre aux deux questions ci-dessus, il impliquerait, en toute généralité, l'interprétation de la syntaxe et de la sémantique du français, ce qui, aujourd'hui, est encore hors de portée.

Signalons enfin que l'introduction, dans une simple *base de données de batailles*, et sous les rubriques «nom» et «date», des données élémentaires «Marignan» et «1515», respectivement, fournirait une représentation adéquate de *l'information*, mais non de la connaissance, contenue dans la phrase. Ces deux données ne comporteraient pas, en effet, les conditions de leur utilisation: ce n'est qu'au travers d'une interprétation du *schéma* de la base, rétablissant les liens existant entre les noms des rubriques (à savoir respectivement «nom» et «date») et les valeurs enregistrées (c'est-à-dire «Marignan» et «1515»), que cette information pourrait être exploitée.

PERSPECTIVES OUVERTES

Comment dès lors expliquer que, jusqu'ici, et en dépit de ses potentialités immenses, cette nouvelle *technologie de la connaissance* n'ait eu qu'un impact limité, et ceci dans des domaines très spécialisés, tels que la prospection minière ou pétrolière, la configuration de systèmes informatiques ou le diagnostic de pannes? Une explication partielle réside dans le fait que la plupart des outils logiciels conçus pour le traitement des connaissances, et en particulier les moteurs d'inférence évoqués ci-dessus, sont écrits dans le langage LISP. Ils requièrent de ce fait un environnement de développement et d'exécution qui leur sont propres. Apparus il y a quelques années, les *processeurs symboliques* sont des machines spécialement conçues pour supporter de tels environnements. Leur coût relativement élevé en a malheureusement limité la diffusion au cas où, comme dans les exemples cités plus haut, l'importance des enjeux économiques justifie à priori la mise en oeuvre de moyens informatiques non-conventionnels.

Cet état de fait pourrait cependant changer bientôt, grâce, une fois encore, à la conjugaison de deux facteurs:

a) l'arrivée, sur le marché, de nouvelles *stations individuelles* (ou *stations de travail*, de l'anglais *workstations*), plus performantes et moins chères; celles-ci pourraient être construites à partir, soit de *microprocesseurs symboliques* (tels que le microExplorer de Texas Instruments), ou, plus vraisemblablement encore, de *processeurs à architecture RISC* (tels que ceux dévoilés dans la station SUN 4), ces derniers étant appelés à supporter, dans un environnement logiciel traditionnel, des applications LISP compilées;

b) la disponibilité, sur ces machines, de *logiciels intégrés pour le traitement des connaissances* tels que KEE, Knowledge Kraft ou G2.

L'interrogation contenue dans le titre de cet article prend ainsi tout son sens:

les logiciels intégrés de traitement des connaissances, tels que KEE, Knowledge Kraft et G2 (ou leurs successeurs), seront-ils demain, pour les stations individuelles, ce que Multiplan, Lotus 1 2 3 ou Framework sont aujourd'hui aux ordinateurs personnels, c'est-à-dire une des clés de leur succès?

LOGICIELS INTEGRES ET TRAITEMENT DES CONNAISSANCES

Sans répondre directement à la question posée ci-dessus, nous nous proposons, dans ce qui suit, de présenter les caractéristiques principales de ce nouveau type de logiciel. Ceci devrait permettre au lecteur d'en apprécier la portée, et, le cas échéant, de déterminer l'intérêt qu'il peut représenter pour lui.

Les logiciels intégrés classiques, tels que Lotus 1 2 3, comportent trois composantes principales, à savoir un *système de traitement de texte*, un *tableur* et un *système de gestion de données*. Il est facile d'associer, à chacune de ces composantes, des fonctionnalités de base dans le domaine du traitement des données.

Exemples

Avec un système de traitement de texte, on peut

- taper un texte
- ajouter, remplacer, déplacer des mots ou des phrases
- définir des mises en page

le tout dans le but d'éditer un document.

Avec un tableur, on peut

- remplir des colonnes de nombres
- définir les opérations à effectuer sur ces données
- représenter graphiquement les résultats

le tout dans le but d'établir la liste des statistiques.

Avec un système de gestion de données, on peut

- enregistrer des fiches comportant différentes rubriques
- définir une question à l'aide de mots-clé
- effectuer une recherche

le tout dans le but de dresser la liste des fiches se rapportant à un sujet donné.

Qu'en est-il, pour leur part, des composantes et fonctionnalités offertes par les logiciels intégrés de traitement des connaissances? En première approximation, de tels logiciels apparaissent essentiellement comme des *systèmes de gestion de base de connaissances.* A ce titre, et par analogie avec les systèmes de gestion de base de données, on pourra, à l'aide de tels systèmes:

- enregistrer la représentation de connaissances;
- définir le profil d'informations manquantes;
- déduire ces informations manquantes.

LE PROBLEME DE LA REPRESENTATION DES CONNAISSANCES

De même que les systèmes de gestion de base de données reposent sur un *modèle de représentation des données*, les différents logiciels de traitement des connaissances se distinguent, au départ, par le choix d'un *modèle de représentation des connaissances.* Parmi les modèles les plus couramment utilisés, on distingue avant tout les modèles *orientés objet.* Dans chacun de ces modèles, on retrouve, sous des terminologies diverses, les principes de base suivants.

Toute parcelle de connaissance se représente comme une *instance*, ou exemple, d'une *classe d'objets* caractérisée par différents *attributs*; chacun de ces attributs peut être, soit de type *déclaratif*, ou descriptif, soit de type *procédural*, c'est-à-dire lié à une utilisation.

Exemple

Aux attributs déclaratifs «nom» et «lieu», hérités d'une simple base de données de batailles pourraient s'ajouter, dans la définition de la classe d'objets correspondante, des attributs procéduraux permettant de répondre à des questions, comme décrit plus haut.

Dans leur forme la plus simple, et la plus courante, les attributs procéduraux attachés aux objets prennent la forme de *règles*, composées chacune de deux parties distinctes:

- les *prémisses*, définissant les conditions d'application de la règle;
- les *conclusions*, indiquant les actions (calcul, déduction, mise à jour, etc.) à entreprendre en cas d'application de la règle.

Exemple

SI objet=bataille & nom= Marignan

ALORS date= 1515.

Comme décrit plus haut, cette règle permet de déduire une date.

Chaque règle de ce type, dite d'*ordre zéro*, détermine une utilisation bien précise de la connaissance représentée par un objet. L'ensemble des objets contenus dans une base de connaissances peut en outre être organisé sous la forme d'une *taxomanie*, ou hiérarchie, de concepts. Grâce au mécanisme d'*héritage* des attributs, chaque objet possèdera, par défaut, les attributs des concepts plus généraux dont il est dérivé. Les règles, dites d'*ordre 1*, définies au niveau des concepts, seront alors applicables à des ensembles d'objets.

Exemple

Si une bataille est définie comme un cas particulier du concept «événement historique», les aspects procéduraux attachés à ce concept seront utilisables pour répondre à des questions sur les batailles.

DEFINITION DU PROFIL DES INFORMATIONS MANQUANTES

Dans le cas le plus simple, le profil d'une information manquante peut être calqué sur les *conclusions* d'une règle, associées à des prémisses bien déterminées.

Exemple

date=? & objet=bataille & nom= Marignan

Ce profil détermine une requête pour obtenir la date de la bataille de Marignan.

Dans les systèmes les plus évolués, et à l'instar de ce que l'on trouve dans certains systèmes de gestion de base de données, cette définition peut s'effectuer à l'aide du langage naturel.

Exemple

- quelle est la date de la bataille de Marignan?
- quel serait le résultat d'exploitation attendu si le budget de publicité était augmenté de 20%, si le taux d'inflation se maintenait à son niveau actuel, et si le cours du dollar se stabilisait autour de Fr. 1,60?

  N.B. Dans ce dernier exemple, le système devra déduire de lui-même la valeur des paramètres manquants, tels que prix de revient et de vente, niveau de la demande, part du marché, etc.

DEDUCTION DES INFORMATIONS MANQUANTES

Si l'on se restreint à l'exploitation de connaissances représentées sous la forme de règles, on peut distinguer deux modes de mise en oeuvre du processus de déduction:

- dans le mode dit de *chaînage avant*, le processus de déduction est amorcé par les objets eux-mêmes; dès que les prémisses d'une règle sont satisfaites par un objet, ses conclusions sont mises en oeuvre automatiquement; celles-ci pouvant avoir pour effet de modifier l'état d'autres objets, de nouvelles règles pourront être déclenchées à leur tour, et ainsi de suite;
- dans le mode dit de *chaînage avant*, le processus de déduction est amorcé par une requête pour une information manquante; si les conclusions d'une règle satisfont le profil d'une information manquante, et si ses prémisses sont satisfaites par les informations disponibles, alors ces conclusions sont admises; sinon, elles deviennent à leur tour des requêtes actives, et ainsi de suite.

Le mécanisme de chaînage avant est particulièrement bien adapté à la modélisation de systèmes impliquant la surveillance d'un grand nombre de variables.

Exemple

Surveillance de transactions financières dans un système de gestion de portefeuille.

Le chaînage arrière, quant à lui, permet, par l'application successive de règles d'ordre un, de sélectionner, dans une classe d'objets, ceux qui jouissent de certaines propriétés.

Exemple

Sélection des placements dans un système de gestion de portefeuille.

Ainsi que le montre les deux exemples ci-dessus, une application peut s'appuyer sur les deux modes de déduction. De ce fait, la plupart des moteurs d'inférence contenus dans les logiciels intégrés fonctionnent, à choix, aussi bien en chaînage avant qu'en chaînage arrière.

UN EXEMPLE DE LOGICIEL INTEGRE: LE SYSTEME G2

Le système **G2**, qui vient d'être acquis par l'Ecole des HEC, est un outil générateur de systèmes experts en temps réel. Représentatif des logiciels intégrés de traitement des connaissances disponibles sur le marché, il comprend de nombreuses facilités pour représenter graphiquement les objets manipulés. Quelques-unes de ses caractéristiques les plus intéressantes sont décrites ci-dessous, dans des termes extraits du descriptif du produit, et qui s'inscrivent dans la ligne de ce qui a été exposé plus haut:

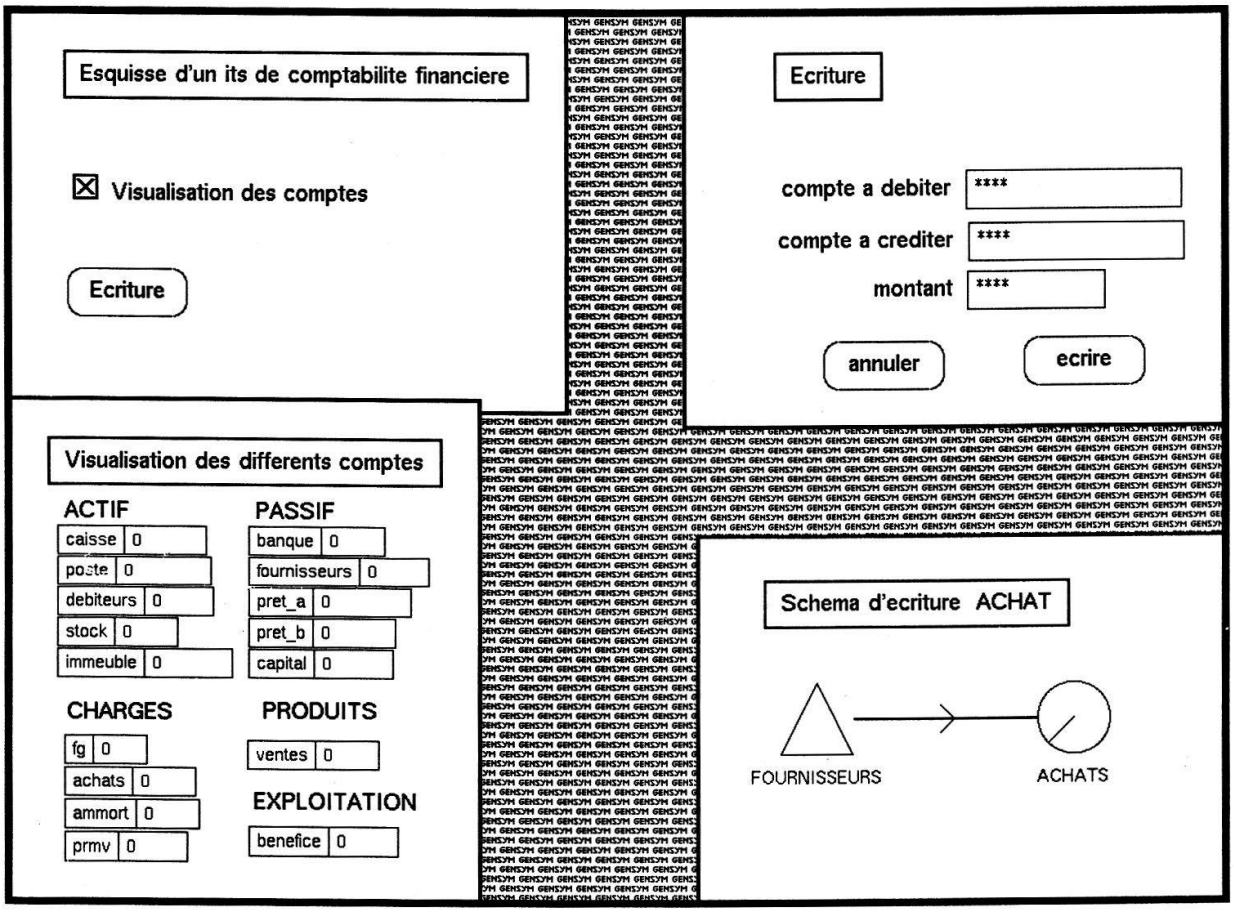

Fig 1: le système attend un jeu d'écritures correspondant à un achat à crédit

«La représentation des connaissances dans G2 est à base d'*objets*, manipulables directement par des icônes. On peut définir des classes d'objets et des hiérarchies de classes (l'héritage portant sur les attributs et sur les icônes). Objets et relations entre objets sont à la fois saisis et représentés sous *forme graphique* dans des schémas. Pour chaque application, on peut construire une *hiérarchie de ces schémas*, à différents niveaux de détail.

Les attributs des objets peuvent être des formules ou des fonctions du type de celles utilisées classiquement dans un tableur. A chaque objet, on peut associer un «modèle dynamique» (au sens de l'automaticien), pour représenter logiquement ou analytiquement son comportement dans le *temps*.

Le langage d'expression des *règles* est proche du langage naturel (cf. le deuxième exemple). Il autorise l'emploi de coefficients de vraisemblance. G2 offre différentes possibilités pour structurer les bases de règles: on peut par exemple affecter à chaque règle une «catégorie», ou distinguer les règles qui s'appliquent aux classes de celles qui s'appliquent aux instances. En dehors des modes classiques de déclenchement en chaînage avant ou arrière, il existe plusieurs autres modes d'activation. C'est ainsi qu'on peut prévoir pour une règle:

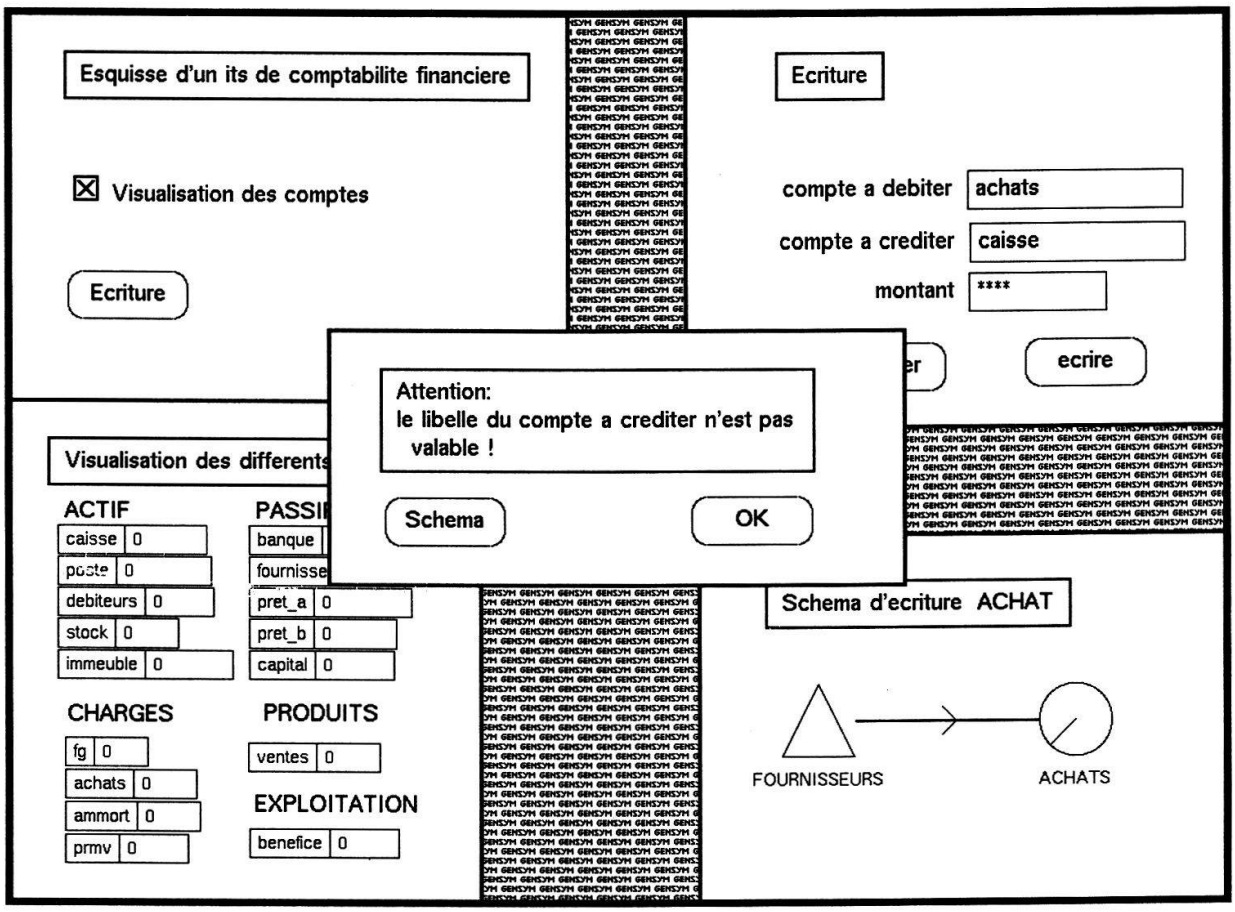

Fig 2: le système refuse une écriture qui ne correspond au schéma affiché

— son activation périodique systématique,

— son activation «en focalisation» quand on décide de centrer le raisonnement sur un contexte ou une entité donnés,

— son activation sur «réflexe» (occurrence d'un événement, arrivée d'une donnée externe).

Mentionnons enfin que le moteur d'inférence est à même de traiter des données étiquetées par un temps et un intervalle de validité.

G2 permet de faire de la *simulation*. Les modes de défaillance et les perturbations peuvent être représentées logiquement ou analytiquement. Des outils graphiques permettent de suivre commodément le déroulement et les résultats de ces simulations. G2 permet aussi la *gestion d'historiques* et leur traitement par des règles».

Les figures données en annexe sont extraites d'une application actuellement en cours de réalisation à l'Ecole des HEC. Cette application, qui vise, à terme, à faciliter la pratique de la

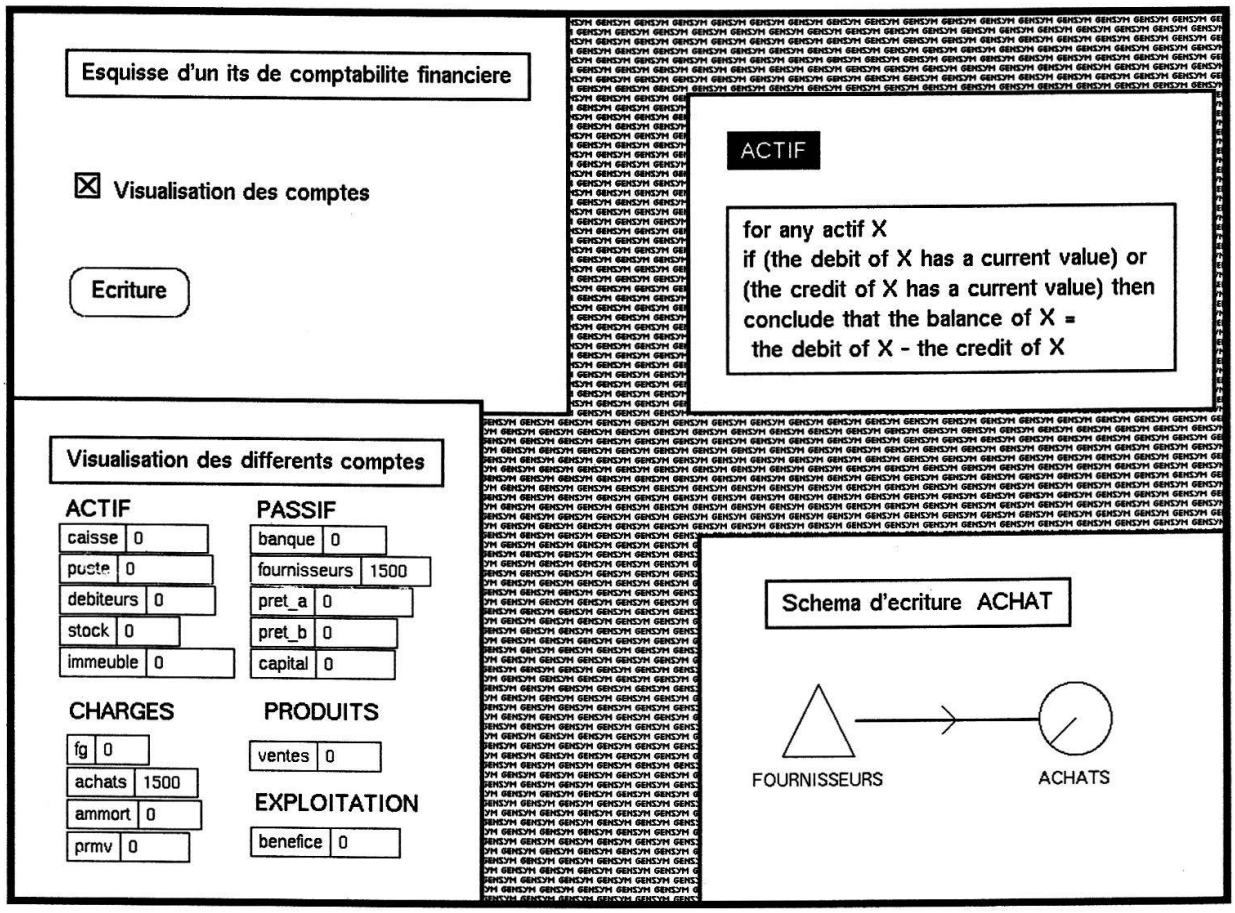

Fig 3: affichage d'une connaissance procédurale

théorie comptable enseignée aux étudiants HEC[2], se présentera sous la forme d'un système expert permettant de vérifier la validité des écritures comptables. En l'état actuel, on se contentera d'illustrer certains points évoqués ci-dessus, à savoir, respectivement, la représentation graphique des objets, conforme à la représentation donnée par la théorie (figure 1), un exemple de mise en oeuvre de la connaissance tirée du modèle correspondant à un achat à crédit (figure 2), ainsi que l'expression d'une règle, reproduisant ici le mécanisme de calcul du solde d'un compte d'actif (figure 3).

CONCLUSIONS

La disponibilité, simultanée, de logiciels intégrés de traitement des connaissances de plus en plus sophistiqués et de stations individuelles de plus en plus puissantes, pourrait bien donner un coup de fouet décisif à la pénétration des techniques de l'intelligence artificielle, et signifier par là-même un nouveau bond en avant de l'informatique individuelle. Les facilités offertes, dans des systèmes comme G2, pour définir la dynamique d'un système au

moyen de règles logiques, jette un pont entre deux domaines fertiles, jusque-là étrangers l'un à l'autre: la rencontre, désormais possible, de l'analyse de systèmes et de la programmation symbolique montrera peut-être la voie à suivre pour réaliser les projets les plus ambitieux de demain.

RÉFÉRENCES

[1]A. Bonnet, *L'intelligence artificielle, Promesses et Réalités,* Interéditions, Paris, 1984

J.L. Laurière, *Intelligence artificielle, résolution de problèmes par l'homme et la machine*, Eyrolles, Paris, 1986

[2]B. Apothéloz et A. Stettler, *Maîtriser l'information comptable*, Presses polytechniques romandes, Lausanne, 1987