PIF_g ou Comment Éditer un Raisonnement dans un STI

Jean-Yves DJAMEN Claude FRASSON Marc KALTENBACH*
Philip OBENSON†
Adil KABBAJ

Université de Montréal, DIRO
Groupe HERON C.P. 6128 Succursale CENTRE-VILLE
Montréal, Québec, H3C 3J8
email: djamen@iro.umontreal.ca

Résumé

Nous décrivons une notation, appelée PIF_g, qui permet à un apprenant d'éditer et d'analyser lui-même son raisonnement par approximations successives. Cette notation a, en particulier, l'avantage de faciliter le diagnostic du raisonnement de l'apprenant dans un système tutoriel intelligent. Le problème de diagnostic, qui relève de la modélisation de l'apprenant, a déjà été décrit comme un sujet de recherche très difficile. Nous montrons comment la notation PIF_g peut aider à résoudre efficacement ce problème. Nous montrons aussi que PIF_g (ou une notation similaire) est un préalable à l'analyse du raisonnement d'un apprenant.

Mots clef : Acquisition des connaissances, Représentation des connaissances, Planification et exécution, Analyse du raisonnement, STI.

*. Bishop's University, Lennoxville
†. Institut de Technologie de l'Information, Douala Cameroun

1 Introduction

L'analyse du raisonnement d'un apprenant est un domaine de recherche très actif dans le cadre des travaux sur les Systèmes Tutoriels Intelligents (STI). Ceci est dû en partie aux difficultés que pose la modélisation de l'apprenant [15, 14, 13]. Le recouvrement [4] et la reconstruction [3] sont les premières approches qui ont été proposées dans la littérature. Très peu de systèmes invitent l'apprenant à participer explicitement au processus de diagnostic de son propre raisonnement. Wenger [17] a relevé, à juste titre, qu'une des solutions les plus significatives au problème de diagnostic serait d'impliquer l'apprenant dans le processus, notamment en lui demandant d'évaluer la compréhension du problème posé et d'expliquer son comportement. Pour y arriver, il est indispensable de permettre à l'apprenant d'avoir une vue réflexive de son propre raisonnement en train de se faire.

Des études en psychologie cognitive [11] ont montré que les états mentaux développés par un apprenant lors d'une session tutorielle comprennent différentes sortes de connaissances, dont en particulier la structure physique de l'appareil, son fonctionnement et son utilisation. Ces connaissances apparaissent davantage dans un contexte de détection et réparation de pannes dans des appareils mécaniques, électroniques ou électro-mécaniques. La structure permettant de les représenter et de les manipuler (analyser) joue, bien entendu, un rôle important dans les buts poursuivis et les stratégies tutorielles adoptées.

L'intérêt porté actuellement par la communauté scientifique sur la représentation des connaissances s'oriente davantage sur les graphes conceptuels (GC) [8], les graphes conceptuels d'acteurs (GCA) [16, 9] ou sur des représentations fondées sur les graphes (Augmented Network Transition - ATN, etc.). En effet, ces représentations offrent une gamme variée d'opérations, pouvant faciliter la conception et l'implantation d'un analyseur de raisonnement. Ces opérations (jointure, appariement, généralisation, spécialisation, projection, contraction, etc.), permettent de faire ressortir les différences entre les graphes fournis par un apprenant et ceux recommandés par un expert du domaine, et de dégager les connaissances attachées aux concepts liés au domaine étudié, etc.

Un analyseur de raisonnement jouissant de cette panoplie d'opérateurs (et de bien d'autres) doit avoir accès (de manière interactive ou non) aux états mentaux de l'apprenant, relatifs aux objectifs qu'on lui a fixé ou qu'il s'est fixé. Or capter le raisonnement d'un apprenant directement dans des structures de types GC, GCA, ATN, etc. n'est pas souhaitable. Ceci est dû au temps d'apprentissage de ces structures, à la complexité des concepts qui les accompagnent et à leur adaptation à la machine.

La plupart des systèmes développés infèrent les aspects du comportement de l'apprenant qui ne sont pas observables. De telles inférences ne garantissent pas que la solution apportée est conforme au problème posé, puisque non fondées sur une connaissance certaine. Nous pensons que la façon la plus naturelle de résoudre le problème de diagnostic du raisonnement de l'apprenant est de donner la possibilité à l'apprenant d'éditer et d'analyser son propre raisonnement. Le système pourra alors l'observer en situation et valider (interactivement ou non) ses actions ou ses intentions, les complèter ou les comprendre, au travers d'une représentation interne

plus efficace.

Il n'est pas toujours facile de trouver une représentation commune pour l'apprenant et le système. Cependant quand deux représentations différentes sont utilisées, il est souhaitable d'avoir un algorithme de passage à l'une ou à l'autre. Plus concrètement, nous pensons à une représentation facile à comprendre et à manipuler par l'apprenant (PIF_g) et pouvant en même temps être transformée dans une représentation facilement manipulable par la machine (GC, GCA, etc.). Il y a donc deux couches dans la représentation des connaissances dans le système sous-jacent: la première est la partie visible (PIF_g) et la deuxième est la partie interne qui sert à l'analyse du raisonnement et qui se base sur les GC.

Depuis un peu plus de 2 ans, nous développons un STI appelé PIF [6, 7], basé sur un modèle (appelé également PIF) constitué de trois mondes P, F et I. Le monde P contient l'objet physique réel manipulé (ou une simulation inspectable). Le monde F est défini en terme de liens causaux et fonctionnels entre objets du monde P. Le monde I permet à un apprenant d'exprimer différentes connaissances sur l'appareil étudié. Les objets manipulés dans ce troisième monde respectent la notation PIF_g.

De façon générale, PIF_g (graphes de connaissances de l'apprenant dans le modèle PIF) dénote un plan, un ensemble de plans ou une notation qui permet de représenter différents états mentaux d'un apprenant sur un objet donné. PIF_g est en même temps un éditeur du raisonnement de l'apprenant dans le modèle PIF.

La composition d'un plan suivant la notation PIF_g permet de faire une première analyse (nous disons *préjugé*) du raisonnement de l'apprenant. Le monde I (contenant un ou plusieurs PIF_g), mis en interaction avec les deux autres mondes (P, F) donne la possibilité à l'apprenant de faire un jugement de valeur sur son plan, i.e éditer son raisonnement et l'analyser. Ceci a pour effet de fournir au système suffisamment d'éléments pour amorcer le processus d'analyse du raisonnement de l'apprenant. La notation PIF_g, par rapport aux autres représentations, apparait donc comme un formalisme naturel permettant de guider l'apprenant sans trop le contraindre.

Dans les sections suivantes, nous présentons d'abord la syntaxe de composition d'un plan PIF_g, commentée des *préjugés* du système sur le raisonnement de l'apprenant. Ces *préjugés* sont utilisés par le système dans le processus d'analyse du raisonnement de l'apprenant. Ensuite nous montrons un exemple de composition de plan PIF_g. Nous montrons, en prélude à l'analyse du raisonnement, le contexte dans lequel les plans peuvent être composés. Enfin nous décrivons l'interfaçage de PIF_g avec d'autres types de représentation de connaissances, notamment les ATN, les GC et les GCA.

2 Syntaxe de composition d'un plan suivant la notation PIF_g

Un plan PIF_g a, linéairement, trois parties principales (figure 1). La première partie (S) regroupe des éléments ou des faits "provocateurs" d'actions: ce sont des *symptômes* associés au problème posé. La deuxième partie (A), est un ensemble

d'actions (effectuées ou à effectuer). A est définie en termes d'*actions/observations* ou de PIF_g; dans ce dernier cas on parle récursion de PIF_g. La troisième partie (H), est le but (ou un ensemble de sous-buts) à atteindre et/ou des hypothèses avancées pour atteindre un but. Un plan PIF_g est de ce fait un agencement de *symptômes*, de couples *(actions, observations)* et d'*hypothèses*; c'est-à-dire une séquence de SAH ou de SPH, P étant un PIF_g.

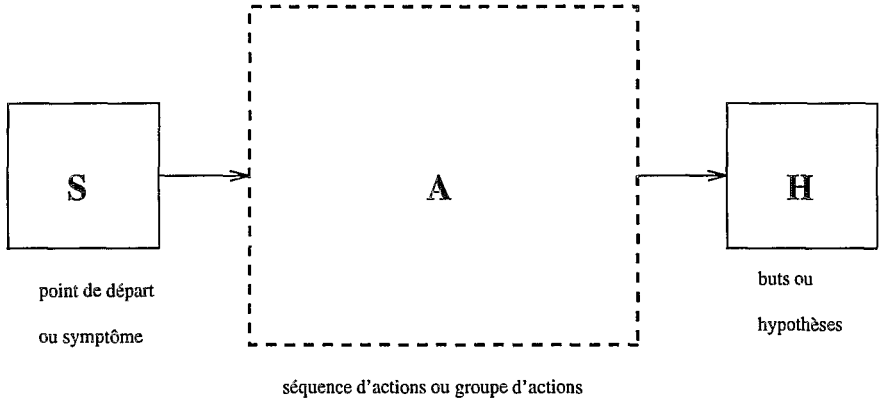


FIG. 1 - *Structure linéaire d'un plan PIF_g*

Plus précisément, un plan suivant la notation PIF_g est un graphe dont les noeuds sont des *symptômes*, des *actions* ou des *hypothèses*; et les arcs, des *observations* (figure 2). Un graphe peut être considéré comme une structure permettant de représenter des connaissances complexes. En fait un graphe en plus d'être un modèle universel pour l'organisation des connaissances est indispensable pour la formulation d'un large spectre de problèmes [1].

Un PIF_g a également une structure dynamique correspondant à son élaboration (composition). La dynamique de la construction d'un plan est un aspect important de la notation PIF_g. Cette construction se fait en plusieurs *étapes* ne suivant pas forcément la linéarité décrite ci-dessus. Chaque étape met le plan en construction dans un *état* précis (voir sous-sections). La structure dynamique est régie par des primitives appelées *opérations*, qui agissent sur les états. Certaines opérations ont pour effet de provoquer un changement d'état et d'autre, au contraire, maintiennent le PIF_g dans le même état. Le graphe de la figure 3 résume l'enchaînement des états et des opérations dans la notation PIF_g. Dans ce graphe, les noeuds représentent les états (ou plus précisément les types d'états) et les arcs représentent les opérations applicables à un état (un arc peut contenir une ou plusieurs opérations). Il y a au total 12 types d'états accessibles dans la syntaxe à l'aide des opérations. Le tableau 1 donne la liste des opérations possibles.

Concrètement, pour décrire les états, nous utilisons des *lettres* (tableau 2) et/ou des *boîtes* (figures 4, 6, 5). Les lettres majuscules et les boîtes pleines (figure 4) désignent un noeud déjà créé (réalisation d'une opération). Les lettres minuscules et les boîtes en pointillés (figure 5) désignent l'attente d'une opération (proposition du système). Une boîte avec un double cadre (figure 6) est utilisée pour désigner le

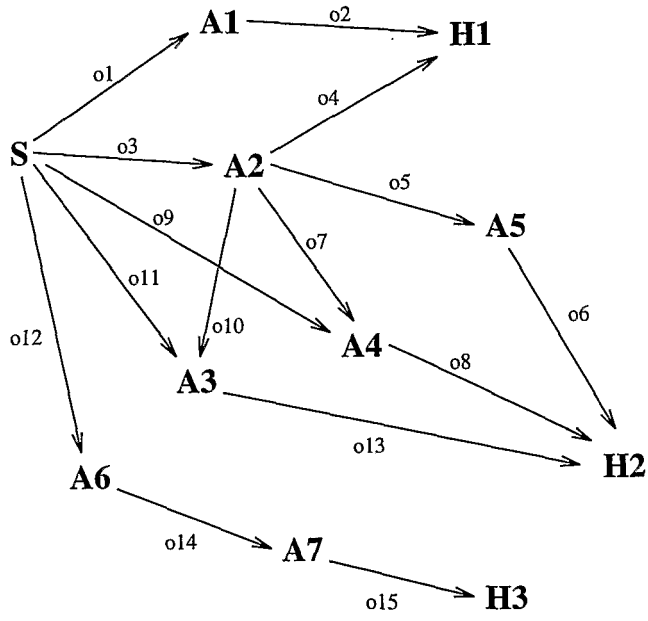


FIG. 2 - *Un plan PIF_g.*

TAB. 1 - *Liste des opérations possibles.*

opérations	*définition*	*littéralement*
bp	commencer un plan	begin plan
hy	faire une hypothèse	hypothesize
ip	insérer un sous-plan	insert plan
sa	mettre une action	set action
sch	choisir l'hypothèse courante	set current hypothesis
sh	mettre une hypothèse	set hypothesis
scn	choisir le noeud courant	set current node
so	mettre une observation	set observation
sp	arrêt de la composition	stop plan

noeud courant ou l'*hypothèse courante* Bien que A, S, H soient des noeuds, *noeud courant* s'applique uniquement à un noeud A ou S. Le terme *hypothèse courante* est utilisé pour les noeuds de type H. Les signes * et + au dessus des lettres indiquent le nombre d'occurrence des objets dans un état. Un signe * désigne une occurrence $(0, n)$ et un signe + désigne $(1, n)$. Par exemple A^+ veut dire qu'il y a au moins un noeud A dans l'état considéré. Comme on le verra dans la suite, les opérations effectuées peuvent faire passer le plan du *mode définition* au *mode résolution* et vice versa. Le *mode définition* correspond à un mode dans lequel le corps du plan n'est pas encore développé; c'est-à-dire un mode dans lequel il n'y a que des hypothèses

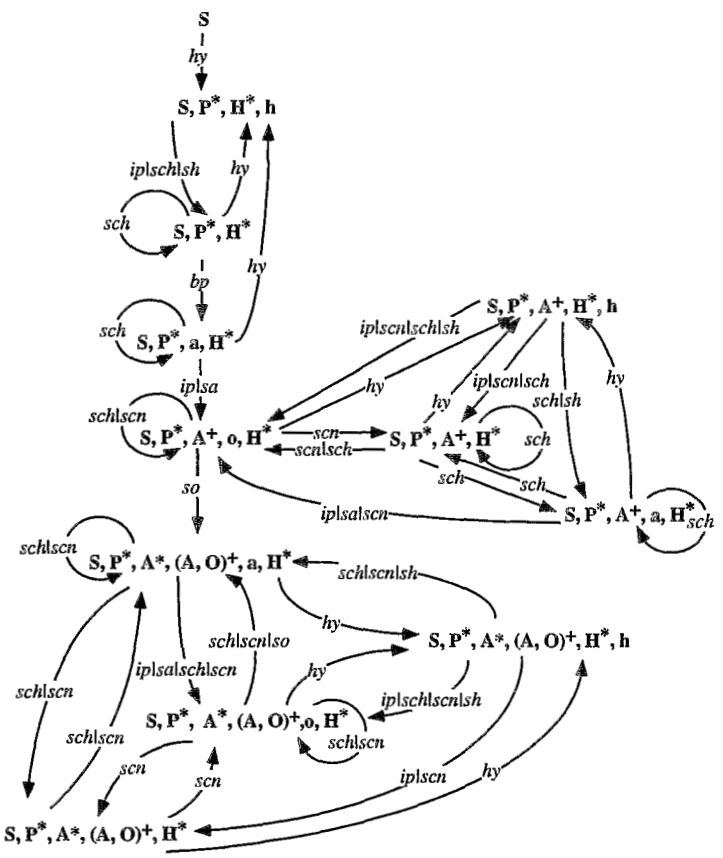


FIG. 3 - *Résumé de la syntaxe de composition d'un plan PIF_g.*

TAB. 2 - *Lettres utilisées et leurs significations.*

lettres	*significations*
A	une action
a	une attente d'action
H	une hypothèse
h	une attente d'hypothèse
O	une observation
o	une attente d'observation
P	un plan
S	un symptôme

qui ont été avancées ou des plans qui ont été inclus. Ceci correspond principalement à la première partie (boîte S) et la troisième partie (boîte H) de la structure linéaire d'un *PIF_g*, et aussi à une abstraction de la deuxième partie (plans in-

FIG. 4 - *Représentation d'une opération déjà effectuée.*

FIG. 5 - *Représentation d'une opération en attente.*

FIG. 6 - *Représentation d'un noeud courant ou d'une hypothèse courante.*

clus). Le *mode résolution* correspond à l'enregistrement des différentes actions et observations.

La syntaxe commence en *mode définition*. L'opération qui permet de passer en *mode résolution* est *bp*. Dans certains cas l'opération *hy* peut faire passer du *mode résolution* au *mode définition*. Dans la figure 3 les états S, SP^*H^*h et SP^*H^* sont en *mode définition* et tous les autres états sont en *mode résolution*.

Les effets des opérations sur les états sont les suivants:

1. **mode définition**
   - *bp* force une attente d'action entre le noeud courant (forcément de type S) et l'hypothèse courante. Il y a passage du *mode définition* au *mode résolution*.
   - *hy* crée un noeud h sous le noeud courant (de type S).
   - *ip* remplace le noeud h concerné par un sous-plan P_i.
   - *sa* (sans effet)
   - *sch* annule tout noeud h existant et transforme l'hypothèse choisie en hypothèse courante.
   - *sh* remplace le noeud h concerné par un noeud H.
   - *scn* (sans effet)
   - *so* (sans effet)
   - *sp* annule toute attente et met fin à la composition du plan.
2. **mode résolution**
   - *bp* (sans effet)
   - *hy* crée un noeud h sous le noeud courant (S ou A) et annule toute attente en cours (noeuds a ou o). S'il n'existe aucun noeud A dans le PIF_g alors il y a changement de mode.
   - *ip* remplace le noeud concerné (h ou a) par un sous-plan P_i et le fait suivre par un lien o, s'il s'agit de a, ou le fait précéder par un o s'il s'agit d'un h.
   - *sa* remplace le noeud a concerné par un noeud A et le fait suivre par un lien o.
   - *sch* annule toute attente (a, h ou o) et transforme l'hypothèse choisie en *hypothèse courante*. Deux cas peuvent alors se poser:
     (a) *le noeud courant est de type A*
     Si entre le *noeud courant* et l'*hypothèse courante* il n'y a aucun noeud,

aucun lien (ou s'il y a seulement un lien non porteur d'observation), cette opération provoque (ou réactive) un lien o entre le *noeud courant* et l'*hypothèse courante*.
Si entre le *noeud courant* et l'*hypothèse courante* il y a un seul lien O, cette opération provoque un noeud a entre O et l'*hypothèse courante*.
Dans les autres cas cette opération provoque (ou fait rester dans) un *état neutre*.

(b) *le noeud courant est de type S*
Si entre le *noeud courant* et l'*hypothèse courante* il n'y a aucun noeud, aucun lien (ou s'il y a un seul lien), cette opération provoque (ou réactive) un noeud a entre le noeud courant et l'*hypothèse courante*.
Dans les autres cas cette opération provoque (ou fait rester dans) un état neutre.

sh transforme le noeud h concerné en noeud H.

scn annule toute attente (noeuds a, h ou o) et transforme le noeud choisi (S ou A) en *noeud courant*. Deux cas peuvent se poser:

(a) *le noeud choisi est de type A*
Si entre le noeud choisi et l'*hypothèse courante* il n'y a aucun noeud, aucun lien (ou si le lien est non porteur d'observation), cette opération provoque un noeud o entre le noeud choisi (devenu *noeud courant*) et l'*hypothèse courante*.
Si entre le noeud choisi et l'hypothèse courante il y a un seul lien O, cette opération provoque un noeud a entre O et l'*hypothèse courante*.
Dans les autres cas cette opération provoque (ou fait rester dans) un état neutre.

(b) *le noeud choisi est de type S*
Si entre le noeud choisi et l'*hypothèse courante* il n'y a aucun noeud, aucun lien (ou s'il y a un lien direct entre les deux), cette opération provoque un noeud a.
Dans tout autre cas cette opération provoque (ou fait rester dans) un état neutre.

so remplace le noeud o en noeud O.

sp annule toute attente et met fin à la composition du plan.

2.1 États en attente d'une hypothèse

Les états en attente d'une hypothèse ont la forme $SP^*A^*(AO)^*H^*h$ (figure 7). Les types d'états contenus dans cette forme générale (SP^*H^*h, $SP^*A^*H^*h$ et $SP^*A^*(AO)^+H^*h$) attentent l'opération *sh*. L'apprenant peut cependant choisir *ip*, *sch* ou *scn* .

ip peut être également considérée comme une opération attendue par le système (elle ne modifie pas vraiment la logique de la construction du plan). Cette opération fait passer à l'état neutre.

sch est une opération exceptionnelle. Par exemple la sélection de H_1 ou de H_2 dans la figure 7 provoque la réactivation (H_1) ou la création (H_2) d'un lien o entre le noeud courant A et l'*hypothèse courante*. Ceci peut vouloir signifier plusieurs choses. D'abord une nouvelle attention portée sur les hypothèses précédemment énoncées par rapport à l'action courante, ou un simple abandon de l'idée d'énoncer une nouvelle hypothèse. Cependant la suite de la composition peut infirmer ou non le sens donné à cette opération.

scn pourrait permettre de se focaliser sur une action du plan en vue de créer d'autres actions ou sur un noeud S en vue de faire une nouvelle hypothèse. Cette opération est considérée comme étant exceptionnelle.

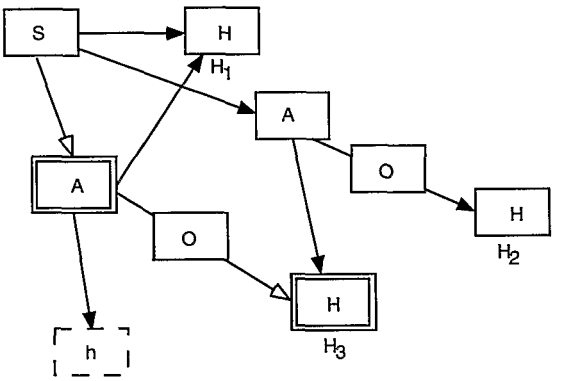


FIG. 7 - *Attente d'hypothèse (Noeud courant de type A).*

2.2 États en attente d'une action

La forme générale des états en attente d'une action est $SP^*A^*(AO)^*H^*a$ (figure 8). Les types d'états (SP^*H^*a, $SP^*A^*H^*a$ et $SP^*A^*(AO)^+H^*a$) attendent l'opération *sa*. Les autres opérations possibles sont *hy, sch, scn.*

hy permet d'avancer une nouvelle hypothèse (différente de celles qui existent) ou précède l'intention d'inclure un plan existant à la suite du noeud courant. La succession des opérations est déterminante dans ce cas.

sch supporte l'intention de faire figurer le *noeud courant* (A ou S) dans la liste des noeuds à visiter jusqu'à l'hypothèse qui sera choisie.

scn peut être une intention de joindre à la liste des actions à effectuer pour l'*hypothèse courante*, le noeud qui sera sélectionné.

2.3 États en attente d'une observation

La forme générale des états en attente d'une action est $SP^*A^*(AO)^*H^*o$ (figure 9). Les types d'états qui s'y trouvent ($SP^*A^+oH^*$ et $SP^*A^*(AO)^+H^*o$) attendent l'opération *so*. Les autres opérations possibles sont *hy, sch, scn.*

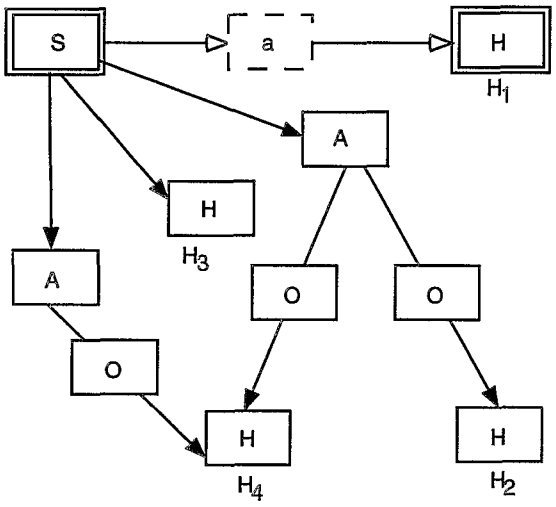


FIG. 8 - *Une attente d'action (noeud courant S).*

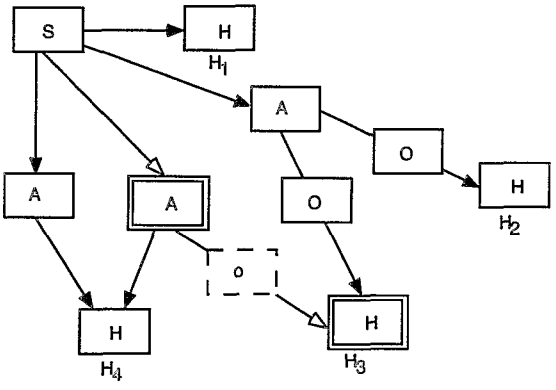


FIG. 9 - *Une attente d'observation.*

2.4 États neutres

Les états neutres S, SP^*H^*, $SP^*A^+H^*$ et $SP^*A^*(AO)^+H^*$ correspondent aux situations où le système n'attend pas une opération particulière. Ils permettent (en général) d'activer les opérations suspendues dans les états précédents à l'aide des opérations *sch, scn.* Les cas particuliers sont le passage du *mode définition* (SP^*H^*) au *mode résolution* (SP^*H^*a) à l'aide de *bp* et l'intention d'avancer une hypothèse (*hy*). La figure 10 montre un exemple d'état neutre.

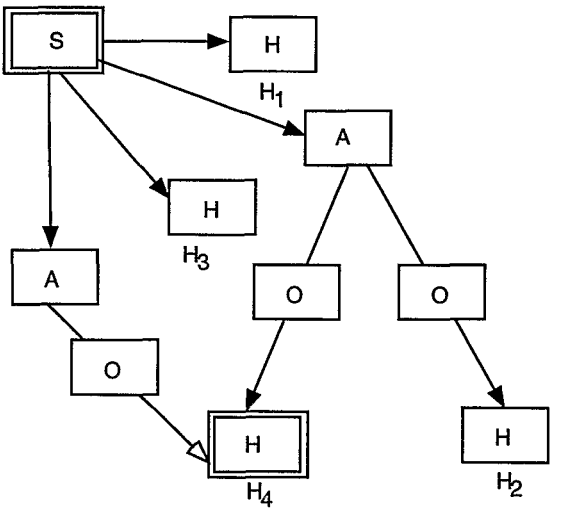


FIG. 10 - *Un état neutre (Noeud courant de type S).*

3 Exemple de composition d'un plan PIF_g

La figure 11 (tirée de [7]) présente la disposition des vues P, I, F dans le système PIF. La session en cours montre, dans la vue I, un plan élaboré par un apprenant suivant la notation PIF_g. Le problème posé est de *trouver pourquoi l'écran affiche une image blanche quand l'ordinateur est mis en fonction.* Dans cette section, nous allons montrer comment ce plan a été composé.

Dans la figure 12 (présentation simplifiée du plan dans I),

S représente le symptôme présenté (*l'écran reste blanc*);

A1 représente l'action *vérifier l'unité J8*;

A2 représente l'action *vérifier l'unité J1*;

O1 représente l'observation *il y a synchronisation*;

O2 représente l'observation *il n'y a pas synchronisation*;

O3 représente l'observation *pas ok*;

O4 représente l'observation *ok*;

H1 représente l'hypothèse *l'unité logique principale est défectueuse*;

H2 représente l'hypothèse *l'alimentation est défectueuse.*

1. Voici un exemple de composition syntaxiquement correcte (Cf. figures ci-dessous dans lesquelles les noeuds courants et les hypothèses courantes sont en gros caractères):
   (a) présentation du symptôme (par le système)
   (b) opération *hy* (intention d'émettre une hypothèse)

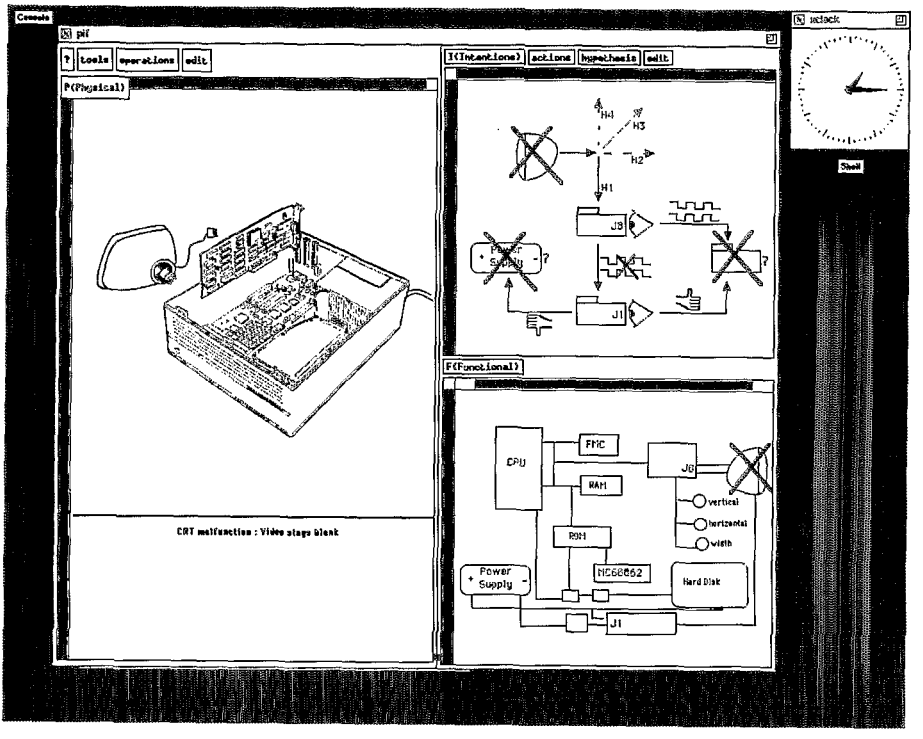


FIG. 11 - *Disposition et contenu des fenêtres du système PIF*

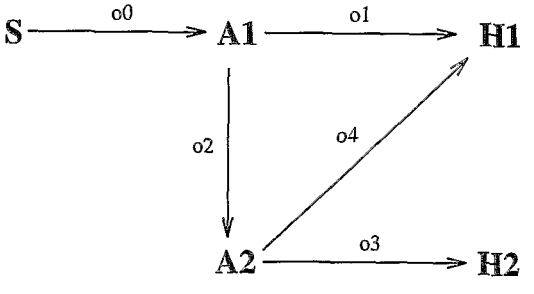


FIG. 12 - *Représentation interne du plan de la vue I.*

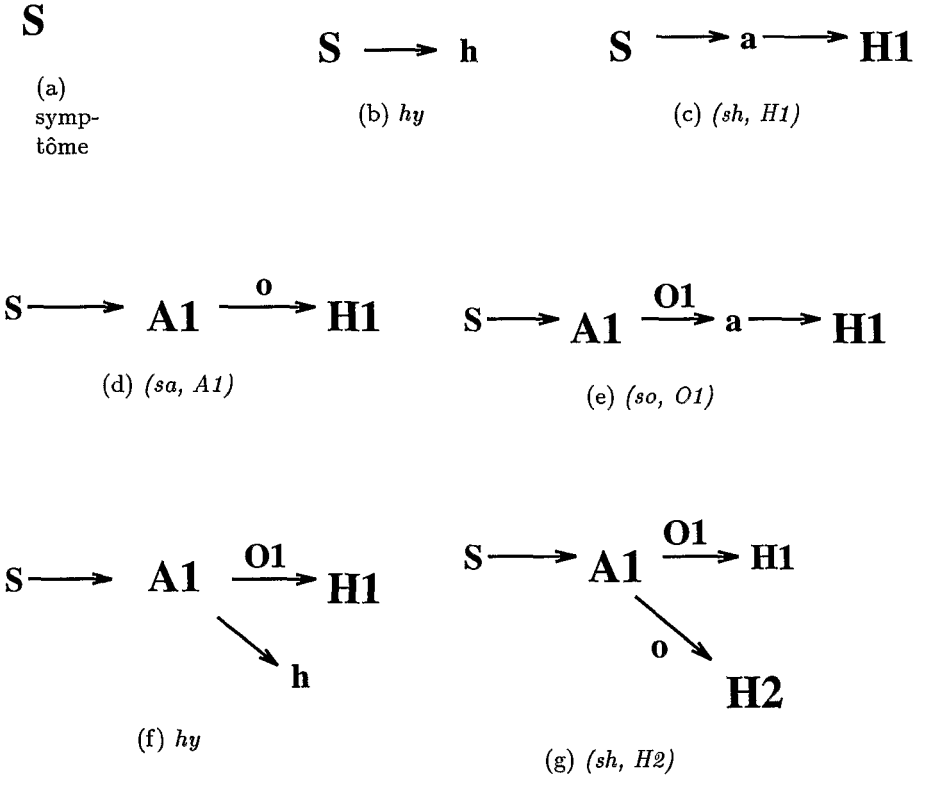
(c) opération *sh* (accomplissement de l'intention avec H1)
(d) opération *sa* (ajout de l'action A1)
(e) opération *so* (ajout de l'observation O1)
(f) opération *hy* (intention d'émettre une nouvelle hypothèse H2)
(g) opération *sh* (accomplissement de l'intention)
(h) opération *so* ajout de l'observation O2)
(i) opération *sa* (ajout de l'action A2)
(j) opération *so* (ajout de l'observation O3)

(k) opération *sch* (changement de l'hypothèse courante, nouvelle hypothèse H1)

(l) opération *so* (ajout de l'observation O4)

(m) opération *sp* (fin de composition de plan)

La représentation interne de cette composition est résumée dans la liste $[S, hy, (sh, H1), (sa, A1), hy, (sh, H2), (so, O2), (sa, A2), (so, O3), (sch, H1), (so, O4), sp]$.



(a) symptôme

(b) *hy*

(c) *(sh, H1)*

(d) *(sa, A1)*

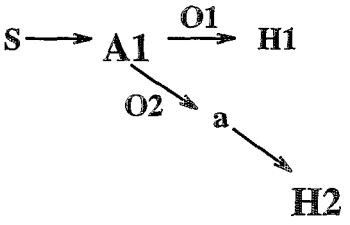
(e) *(so, O1)*

(f) *hy*

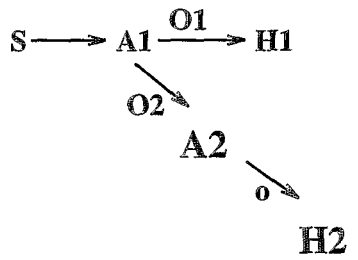
(g) *(sh, H2)*

Dans la composition ci-dessus, nous avons pris le cas d'un apprenant qui se base prioritairement sur H1 (pour des raisons qui peuvent être justifiées dans une session tutorielle).

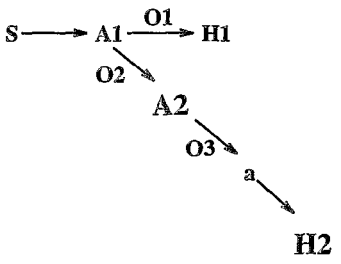
2. Voici une autre composition donnant la priorité à H2.
   - présentation du symptôme
   - opération *hy* (intention d'émettre l'hypothèse H2)
   - opération *sh* (accomplissement de l'intention)
   - opération *sa* (ajout de l'action A1)
   - opération *so* (ajout de l'observation O2)



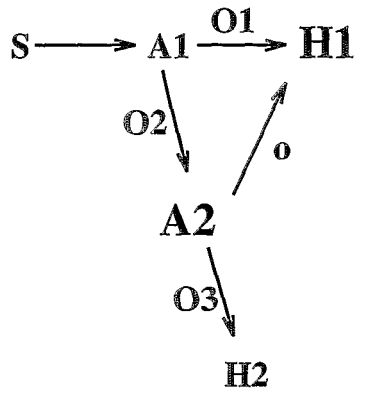
(h) *(so, O2)*



(i) *(sa, A2)*



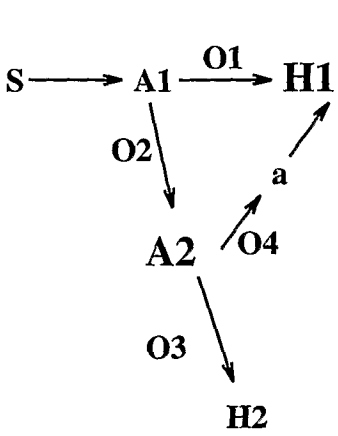
(j) *(so, O3)*



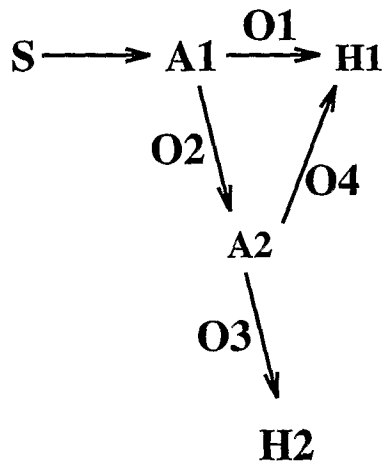
(k) *(sch, H1)*

- opération *sa* (ajout de l'action A2)
- opération *so* (ajout de l'observation O3)
- opération *hy* (intention d'émettre une nouvelle hypothèse, H1)
- opération *sh* (accomplissement de l'intention)
- opération *so* ajout d'une nouvelle observation, O4)
- opération *scn* changement du noeud courant
- opération *so* (ajout d'une nouvelle observation, O1)
- opération *sp* (fin de composition de plan)

Bien que ne faisant pas l'objet de cet article, notons que dans le système PIF l'édition d'un plan (dans la vue I) ne passe pas obligatoirement par les étapes énumérées dans la syntaxe ci-dessus. Il est en effet possible à un apprenant d'utiliser l'éditeur général (ne faisant pas appel à la notation PIF_g) pour composer un plan de détection (et/ou dans certains cas de réparation) de pannes. Dans ce cas il lui est proposé une palette contenant des boites représentant les noeuds de type S, A ou H, et une flèche représentant les observations. Il lui est également proposé un



(l) *(so, O4)*



(m) *sp*

ensemble d'outils permettant de sélectionner des boites et des arcs pour former un graphe. Cet éditeur est généré automatiquement à l'aide d'un *éditeur générique de graphe généralisé* [10].

4 Vers une analyse du raisonnement de l'apprenant

Le système *PIF* implique l'apprenant dans le processus d'analyse de son raisonnement en se basant sur la notation *PIF_g* et de deux modes d'interaction appelés *exécution d'abord* et *planification d'abord.*

L'exécution est composée d'une *exécution libre* dans les vues P et F (mode 1) et d'une *exécution interactive* basée en plus sur la vue I (mode 2). La planification est également composée d'une *planification libre* (mode 3) et d'une *planification interactive* [7] (mode 4).

Comme le montrent les figures 15 et 16, l'apprenant est impliqué directement dans le processus d'analyse de son raisonnement. En effet, à travers la justification de ses actions, ses hypothèses et les buts visés, il fournit au système des informations sur son état mental qui ne sont pas observables autrement. L'implication de l'apprenant dans le processus d'analyse du raisonnement a pour but de faciliter l'analyse que fera le système. De plus elle permet de valider les informations fournies par l'apprenant durant les sessions d'apprentissage.

Pour illustrer tout ceci, nous allons prendre l'exemple d'un problème S qui est posé à l'apprenant et dont la solution de l'expert apparaît en figure 13. (En général dans les STI, les solutions de l'expert doivent tenir compte de l'apprenant (novice, intermédiaire, expert, etc.). Dans cet article nous n'avons pas tenu compte du niveau de l'apprenant. Expert ici veut donc dire auteur et non niveau expert

ou autre.) Les détails des noeuds pouvant être complexes, nous nous limitons à une représentation simple telle que définie dans la figure 14 (les A_i contiennent une séquence linéaire d'actions élémentaires. Par exemple A1 est composé de a1 et a2; A2 de a1, a3, a4 et a5; etc.). Supposons que le problème est posé à l'apprenant à l'instant t_0 et que sa solution apparaît à l'instant t_1. Il faut imaginer un *super stratège pédagogue* qui sait comment amener l'apprenant à passer d'un mode à un autre. Par exemple, un débutant serait porté à démontrer des connaissances opérationnelles (exécution d'abord: modes 1 et 2). Un des rôles du *super stratège pédagogue* consiste à encourager l'apprenant à exhiber des connaissances plus théoriques (planification d'abord: modes 3 et 4), ou dans d'autres cas de passer des connaissances théoriques vers des connaissances pratiques.

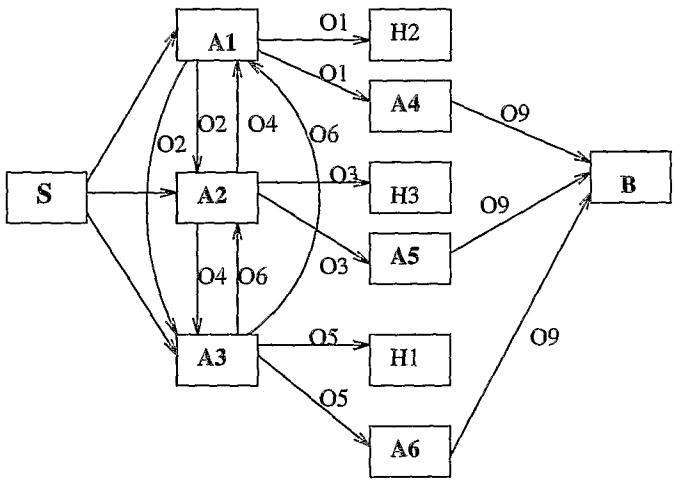


FIG. 13 - *La solution de l'auteur pour le problème S.*

Mode 1: L'apprenant exécute librement ses actions

L'apprenant fait des manipulations (exécution) dans P et F (figure 15) dans le but de résoudre le problème posé. Ces manipulations sont répertoriées dans une trace (trace d'exécution). À l'instant t_1 la trace $a_1, a_2, ..., a_n$ obtenue correspond aux actions (élémentaires) effectuées. (La trace peut être considérée comme étant déjà formée ou en train de se former (action par action), dépendant du degré de contrôle du système.) Le passage de a_i vers a_{i+1} n'est pas visible à ce stade; seul l'apprenant sait ce qu'il a observé. De plus les sous-buts ou buts visés à travers une séquence de a_i ne sont pas accessibles non plus par le système. Une déduction prématurée consisterait à se baser sur la solution de l'expert (figure 13) pour faire ressortir les informations manquantes. Par exemple supposons que la suite d'actions effectuées par l'apprenant est $a_1, a_2, a_7, a_3, a_8, a_{11}, a_5, a_{13}, a_{12}$. Le système pourrait, par application d'une opération d'extraction, reconnaitre l'exécution de la séquence A_1, A_5, menant à B (le but), c'est-à-dire détecter le chemin $S, A_1, O_2, A_2, O_3, A_5, O_9, B$. Ceci ajoute les observations O_2, O_3, O_9 et l'action A_2

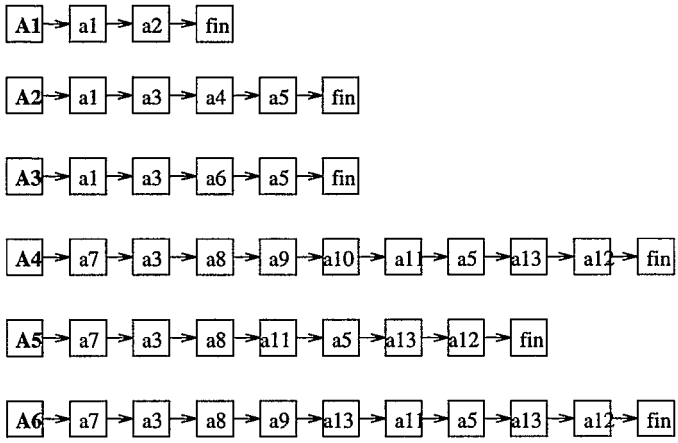


FIG. 14 - *Détail des actions proposées dans la solution de l'expert.*

dans le modèle de l'apprenant. Cependant le système ne devrait pas se servir de ces informations immédiatement pour l'analyse du raisonnement, car rien ne prouve que l'apprenant a effectivement construit ces états mentaux.

Après exécution donc, le système propose à l'apprenant de passer dans I pour éditer sa trace $(a_1, a_2, a_7, a_3, a_8, a_{11}, a_5, a_{13}, a_{12})$. Si pendant l'édition l'apprenant est incapable de fournir les états mentaux non exhibés, le système peut se mettre en mode *coaching* et se servir de la solution de l'expert (figure 13). (Pendant l'édition, le film des actions de l'apprenant est rejoué, pour lui permettre d'accéder à ses propres connaissances. Une technique que nous comptons expérimenter consiste à ne pas rejouer le film dans certains cas afin de noter le degré de mémorisation des actions de l'apprenant.)

A la fin de l'édition on obtient un plan PIF_g qui est en fait la trace de départ, augmentée des buts, sous-buts visés et des observations entre a_i. Il est possible à ce stade de faire une comparaison du plan obtenu avec la solution de l'expert. Cependant advenant un non appariement des deux solutions, celle de l'apprenant est revue sous un autre angle. En fait, il y a vérification de concordance entre les o_i ajoutées et l'état réel de la machine; et pour cela le système se base sur la conceptualisation de la structure physique et du fonctionnement de l'objet étudié (non traitée ici).

Mode 2: L'apprenant exécute et énonce partiellement ses intentions

L'apprenant fait quelques manipulations dans P et F (figure 15) et la séquence d'action qui en résulte est affichée en même temps dans I. L'apprenant peut alors interrompre momentanément l'exécution pour entrer directement les observations entre les a_i. Il peut également interrompre l'édition pour continuer l'exécution. Cette navigation est permise jusqu'à résolution du problème (instant t_1). En plus des observations qu'il doit préciser, l'apprenant peut ajouter des actions dans le

nouveau plan; ce qui introduit un nouveau type d'actions (disons a_j) non encore exécutées. À la fin de la résolution du problème, le système invite l'apprenant à passer dans I pour éditer la trace finale. Puis il y a exécution finale et planification comme dans le mode précédent.

Ce mode est différent du mode exécution libre des actions. D'abord parce qu'il autorise l'édition de la trace du raisonnement avant la fin de la résolution du problème, ensuite parce que l'apprenant a l'occasion de spécifier directement les observations entre les a_i et ajouter des a_j qui seront contrôlées lors de l'exécution finale. La pratique des deux modes peut permettre de définir une nouvelle classe d'apprenants capables de mémoriser (à long terme) des séquences d'actions et les enchainements entre elles.

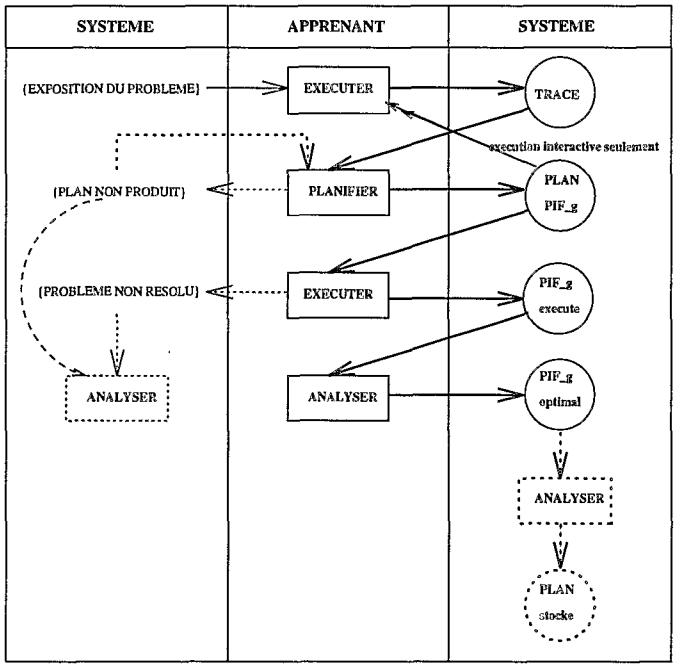


FIG. 15 - *Exécution d'abord.*

Mode 3: L'apprenant planifie librement ses actions

L'apprenant élabore un plan dans I à l'aide de la notation PIF_g (figure 16). Ce plan correspond à un ensemble d'actions (pas nécessairement linéaires) que l'apprenant compte effectuer pour résoudre le problème qui est posé. Il contient en outre des hypothèses avancées et des sous-buts que l'apprenant désire atteindre. Nous avons vu plus haut comment le système fait une première analyse (appelée *préjugé*) des étapes de composition d'un plan de l'apprenant. En effet, à travers l'utilisation (par l'apprenant) des primitives définies dans la syntaxe de composition d'un PIF_g, le système reconnaît les intentions de l'apprenant.

À la fin de la construction de la solution, le système demande à l'apprenant d'exécuter son plan dans P et/ou F, et de l'analyser par la suite. En fait, pendant l'exécution, si l'apprenant effectue une action qui n'apparaît pas dans le plan, le système lui demande de changer d'action ou alors de réviser le plan. Tout compte fait, le système ne fait l'analyse du raisonnement de l'apprenant qu'après analyse faite par l'apprenant lui-même.

Ici, à l'instant t_1 le plan contient des a_i et des o_j non nécessairement élémentaires. Le passage à l'exécution permet de vérifier la faisabilité des actions énoncées et des observations attendues. (Les modes 3 et 4 font appel à l'expansion et à la contraction des types, du fait que l'apprenant peut introduire des actions non élémentaires. Par exemple l'action *vérifier l'unité J1* contient des actions de préparation telles que *enlever le couvercle de l'appareil, etc*).

Mode 4: L'apprenant planifie et exécute partiellement ses actions

L'apprenant fait une ébauche de son plan dans I comme précédemment et a la possibilité de vérifier quelques actions sur la machine réelle, par interruption momentannée de l'élaboration de la solution (figure 16). Contrairement au mode précédent, le système peut intervenir dans la planification ou lors de l'exécution avant la fin de la résolution du problème. Cette intervention inclut une obligation d'inclure, dans le plan, une action qui ne s'y trouvait pas et qui a été testée par l'apprenant dans P ou F avant t_1. À la fin de la planification interactive, le système demande à l'apprenant d'analyser son propre raisonnement.

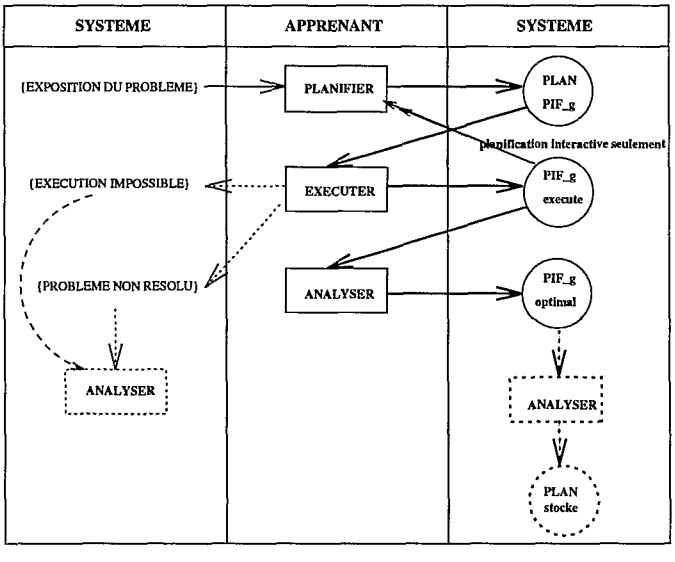


FIG. 16 - *Planification d'abord.*

5 Interfaçage de la notation PIF_g avec les autres représentations

Très peu de systèmes tutoriels existent, qui forcent l'apprenant à avoir une attitude réflexive face à un problème. (Par attitude réflexive, nous entendons la possibilité qui est donnée à l'apprenant d'énoncer et de manipuler (transformer) son raisonnement lors de la résolution d'un problème.) Les systèmes développés ont plutôt cherché à représenter directement les connaissances de l'apprenant sous une forme rapidement manipulable par la machine.

Nous pensons qu'une étape manque dans cette démarche, notamment celle qui permet à l'apprenant d'exprimer ses connaissances sous une forme visible et manipulable, non seulement par le système, mais surtout et avant tout par l'apprenant. Puisqu'il n'est pas toujours facile de trouver une représentation directement manipulable par l'apprenant et le système, il s'agit de privilégier la représentation facilitant la captation directe du raisonnement de l'apprenant, quitte à la transformer ensuite dans une des notations mieux adaptées aux manipulations formelles. Ayant proposé PIF_g qui permet de capter directement le raisonnement de l'apprenant, nous avons relevé pour cette section quelques représentations qui nous paraissent propices pour une manipulation adéquate des connaissances dans la machine. Nous présentons d'abord un algorithme qui permet de passer d'un plan PIF_g à un GC ou un GCA et un autre qui permet de passer à un ATN.

Passage aux graphes conceptuels

Pour la manipulation formelle des connaissances, nous faisons appel aux GC et GCA. En fait, un PIF_g est déjà une forme simplifiée de GC. Cependant dans un soucis d'uniformisation et de clarification, nous avons écrit un algorithme qui permet de passer explicitement d'un plan PIF_g à un GC ou un GCA. Nos hypothèses de départ sont vérifiées par les structures obtenues.

Les représentations sous forme GC s'appuient sur les concepts (représentés par des rectangles) et des relations conceptuels (représentées par un arc avec label ou un cercle). Les GCA utilisent en plus des concepts et des relations entre concepts, des acteurs (représentés par des losanges). Un concept est composé du type du concept et du référent du concept [16].

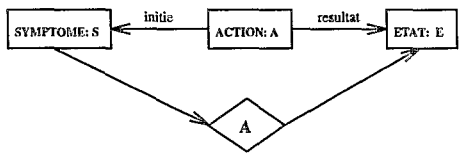


FIG. 17 - *Schéma pour symptôme et action.*

La transformation d'un PIF_g en un GC se fait de la manière suivante:

- transformer tous les noeuds S en concept SYMPTÔME ayant pour référent le contenu du noeud;

- transformer tous les noeuds A en concept ACTION ayant pour référent le contenu du noeud;
- transformer tous les noeuds H en concept HYPOTHESE ayant pour référent le contenu du noeud;
- transformer tout schéma (symptôme, action) par: *un SYMPTÔME S initie une ACTION A dont le résultat est un ETAT E*;
- transformer tout schéma (action, observation, action) par: *une OBSERVATION O qui satisfait un ETAT E1 initie une ACTION A dont le résultat est ETAT E2*;
- transformer tout schéma (action, observation, hypothèse) par: *si OBSERVATION O satisfait un ETAT E1, alors il y a suggestion de l'HYPOTHESE H.*

Pour le passage en GCA, il faut ajouter à l'algorithme ci-dessus ce qui suit:

- tout schéma (symptôme, action) donne lieu à un acteur A (figure 17);
- tout schéma (action, observation, action) donne deux acteurs A et UNIFIER et un concept BOOLEEN (figure 18);
- tout schéma (action, observation, hypothèse) donne deux acteurs UNIFIER et EMETTRE et un concept BOOLEEN (figure 19).

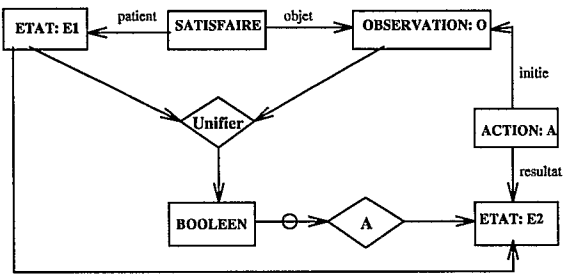


FIG. 18 - *Schéma pour action vers action.*

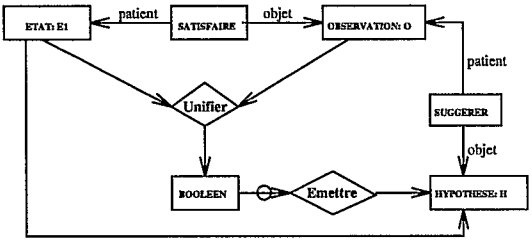


FIG. 19 - *Schéma pour action vers hypothèse.*

En appliquant les deux algorithmes ci-dessus, l'exemple de la figure 12 est transformé tel que l'indique la figure 20. Cette figure combine les GC (boites et flèches pleines uniquement) et les GCA (ensemble du graphe).

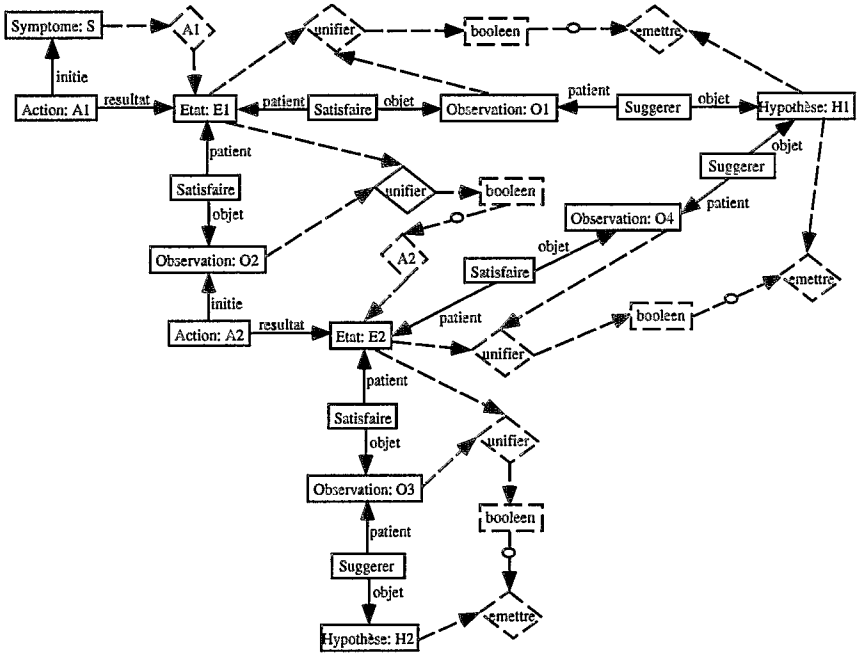


FIG. 20 - *Un graphe conceptuel d'acteurs.*

Passage aux ATN

Les ATN sont des graphes dont les noeuds sont des états (état de départ ou état cible) et dont les arcs forment un couple (condition, action). Nous représentons une condition ou une action vide par *nil*. L'algorithme de transformation a l'allure suivante:

```
PIF_gVersATN(NUMERO, ETAT)
mettre etat de depart genere a partir de NUMERO dans Ed
mettre ETAT dans val(Ed)
pour tout arc partant de ETAT faire
debut
        mettre contenu-de-l-arc dans C
        si cible-de-l-arc est une hypothese
        debut
                attribuer un numero s a l'etat suivant Es
                mettre cible-de-l-arc dans val(Es)
                mettre nil dans A
                tracer(Ed, Es, C, A)
        fin
        sinon  debut
                        attribuer un numero s a l'etat suivant Es
                        mettre nil dans val(Es)
```

mettre cible-de-l-arc dans A
tracer(Ed, Es, C, A)
planVersATN(s, Es)
fin
fin

Le programme suivant, écrit en IF/Prolog version 4.1.8 et testé sur INDIGO IRIS 4.0.5 met en oeuvre cet algorithme (le jeu d'essai est celui de la figure 2):

```
%%%%%%%
%%% programme qui transforme un plan PIF_g en ATN
%%% 20 nov. 1993 *** (c) DJAHEN ***
%%% pour lancer, faire pifg2atn et voir resultat dans sortie.
%%%%%%%

pifg2atn :- tell(sortie),planVersATN(0,s), nettoyer.

% jeu d'essai
% pifg(Etat_de_depart, Etat_suivant, Arc)
pifg(s, a1, o1).   pifg(s, a2, o3).   pifg(s, a3, o11).  pifg(s, a4, o9).  pifg(s, a6, o12).  pifg(a1, h1, o2).
pifg(a2, h1, o4).  pifg(a2, a5, o5).  pifg(a2, a4, o7).  pifg(a2, a3, o10).  pifg(a3, h2, o13). pifg(a4, h2, o8).
pifg(a5, h2, o6).  pifg(a6, a7, o14).  pifg(a7, h3, o15).

% hypo(Hypothese)
%les  hypotheses du plan
hypo(h1).  hypo(h2).  hypo(h3).  hypo(h4).

planVersATN(N, E) :- attEtat(N, Ed), insererValEtat(Ed, E),
bagof([Cible, C], pifg(E, Cible, C), ListeCible),
touslesArcs(ListeCible, Ed).

touslesArcs([[Cible, C]|R], Ed):- traitement(Ed, Cible, C), touslesArcs(R, Ed).
touslesArcs([],_).

traitement(Ed, Cible, C) :- hypo(Cible), !, trtHypo(Ed, Cible, C).

% traitement d'une action
traitement(Ed, Cible, C) :- trtAction(Ed, Cible, C).

trtAction(Ed, Cible, C):- clause(valEtat(E, Cible)), !, write('('),write([Ed, E, C, Cible]), write(')'), nl.
trtAction(Ed, Cible, C):- attNumeroEtat(N), attEtat(N, Es), insererValEtat(Es, Cible), write('('),
write([Ed, Es, C, Cible]), write(')'), nl, planVersATN(N, Cible).

% traitement d'une hypothese
trtHypo(Ed, Cible, C) :- clause(valEtat(E,Cible)), !,  write('('), write([Ed, E, C, nil]), write(')'), nl.
trtHypo(Ed, Cible, C) :- attNumeroEtat(N), attEtat(N, Es), insererValEtat(Es, Cible),  write('('),
write([Ed, Es, C, nil]), write(')'), nl.

% attribution d'un etat
attEtat(N, E) :- append([e], [N], X), concat(X, E).

% attribution d'un numero d'etat (atn)
attNumeroEtat(NN):- retract(seqATN(N)), !, NN is N + 1, assertz(seqATN(NN)).
attNumeroEtat(1) :- assertz(seqATN(1)).

insererValEtat(Ed, _):- clause(valEtat(Ed, _)), !.
insererValEtat(Ed, E):- assertz(valEtat(Ed, E)), write('valeurEtat('),
write(Ed), write(','), write(E), write(')'), nl.

% nettoyage
nettoyer :- enlever(valEtat(_,_)).

enlever(X):- retract(X), enlever(X).

%%% resultat pris dans le fichier sortie
% atn([Etat_depart, Etat_arrivee, Condition, Action]).
atn([e0,e1,o1,a1]).   atn([e1,e2,o2,nil]).  atn([e0,e3,o3,a2]).
atn([e3,e2,o4,nil]).  atn([e3,e4,o5,a5]).   atn([e4,e5,o6,nil]).
atn([e3,e6,o7,a4]).   atn([e6,e5,o8,nil]).  atn([e3,e7,o10,a3]).
atn([e7,e5,o13,nil]). atn([e0,e7,o11,a3]).  atn([e0,e6,o9,a4]).
atn([e0,e8,o12,a6]).  atn([e8,e9,o14,a7]).

% valeurEtat(Etat, Valeur)
```

```
valeurEtat(e0,s).  valeurEtat(e1,a1).  valeurEtat(e2,h1).
valeurEtat(e3,a2).  valeurEtat(e4,a5).  valeurEtat(e5,h2).  valeurEtat(e6,a4).
valeurEtat(e7,a3).  valeurEtat(e8,a6).  valeurEtat(e9,a7).  valeurEtat(e10,h3).
```

6 Conclusion

Nous avons présenté *PIF_g*, une notation qui permet de représenter différentes connaissances de l'apprenant en situation de résolution de problème. WUSOR [4], BIP [2] sont les premières recherches en STI qui ont tenté (et qui n'ont pas réussi) de faire participer l'apprenant dans le processus de diagnotic. Les recherches qui ont suivi et qui ont apporté une amélioration sensible au problème de diagnostic, grâce à ce qui est communément appelé *diagnostic interactif* sont, entre autres, ACE [15] (pour la compréhension des explications du raisonnement en termes de traces internes), et GUIDON [5] (pour le sondage des connaissances en demandant la justification des prédictions ou des hypothèses). Nous pensons que seul le système GIL (Graphical Instruction in Lisp) [12] semble se rapprocher des solutions proposées dans le système *PIF* par l'entremise de la notation *PIF_g*. GIL a été concu pour explorer la construction des explications à partir des connaissances de résolution de problème et l'utilisation des représentations visuelles dans la résolution des problèmes. Cependant, le but principal de GIL est de donner une alternative au mode texte, grâce à une représentation graphique. *PIF_g*, en plus d'incorporer tous les aspects multimédia, donne la possibilté à l'apprenant de visualiser son raisonnement et de le modifier sans l'intervention du système. Aussi, une différence assez significative entre GIL et *PIF_g* semble se trouver au niveau de la structure des éléments du graphe. Les noeuds et les liens dans le graphe GIL sont très simples et ne peuvent pas supporter une structure complexe, par exemple un graphe lui-même.

La notation *PIF_g* est une composante essentielle du système *PIF* qui est en cours d'implantation. Nous prévoyons des expériences avec des infirmières utilisant une pompe à infuser en soins intensifs.

Références

[1] Aho, A. V., and Ullman, J. D. *Foundations of computer science.* W. H. Freeman and Company, 1992.

[2] Barr, A., and Atkinson, R. C. The computer as tutorial laboratory: the stanford BIP project. *Int Jrnl Man-Machine Studies 8* (1976), 567–596.

[3] Brown, J. S., Collins, A., and Harris, G. Artificial intelligence and learning strategies. In *Learning strategies*, H. O'Neil, Ed. Academic Press, 1978.

[4] Carr, B., and Goldstein, I. P. *Overlays: a theory of modeling for computer-aided instruction.* Massachusetts Institute of Technology, CAmbridge, Massachusetts, 1977.

[5] CLANCEY, W. J. Tutoring rules for guiding a case method dialog. In *Intelligent Tutoring Systems*, D. H. Sleeman and J. S. Brown, Eds. Academic Press, London, 1979.

[6] DJAMEN, J.-Y., FRASSON, C., KALTENBACH, M., AND GECSEI, J. The PIF system or solving a problem from multiple perspectives. In *EXPERSYS '92*, F. Attia, S. Flory, S. Hashemi, G. Gouardères, and J. P. Marciano, Eds. I.I.T.T. International, Paris, 1992, pp. 37–43.

[7] DJAMEN, J.-Y., KALTENBACH, M., AND FRASSON, C. An interactive planning and learning system. In *Artificial intelligence in education* (Edinburgh, Scotland, 1993), AI-ED '93.

[8] Workshop on PEIRCE: a conceptual graph workbench, 1993.

[9] KABBAJ, A. Toward a conceptual actor language. In *Knowledge representation with conceptual graphs*, G. Mineau, B. Moulin, and J. Sowa, Eds. Springer Verlag, Québec, 1993.

[10] KABBAJ, A., AND DJAMEN, J.-Y. Un éditeur générique de graphe généralisé, 1994.

[11] KIERAS, D. E., AND BOVAIR, S. The role of a mental model in learning to operate a device. *Cognitive Science 8* (1984).

[12] REISER, B., RANNEY, M., LOVETT, M., AND KIMBERG, D. *Facilitating Students Reasoning with Causal Explanations and Visual representations.* 4th AI-ED conference. IOS, Amsterdam, 1989.

[13] SELF, J. A. Bypassing the intractable problem of student modelling. In *Intelligent Tutoring Systems* (Montréal, 1988), ITS-88, pp. 18–24.

[14] SLEEMAN, D. H. A user-modelling front-end subsystem. *Int Jrnl Man Machine Studies 23* (1984).

[15] SLEEMAN, D. H., AND HENDLEY, R. J. ACE: a system which analyses complex explanations. *Int Jrnl Man Machine Studies 11* (1979).

[16] SOWA, J. *Conceptual Structures. Information processing in mind and machine.* Addison Wesley, 1984.

[17] WENGER, E. *Artificial Intelligence and Tutoring Systems.* Morgan kaufmann publishers, Inc, Los Altos, 1987.