Experts, contextes et événements

Description du système Prof'Expert pour le programmeur

L.-O. Pochon, C. von Siebenthal

Association ABORD, 1994

Table des matières

Introduction

Prof'Expert est un système destiné à l'EAO. Son but est de permettre la mise sur pied de séquences de tutorat en ne s'occupant que des données. Le contrôle général, la disposition des fenêtres, les évaluations se déroulent de façon automatique selon un certain nombre de scénarios préétablis. Le système est subdivisé en plusieurs programmes. La version modulaire est constituée de parties qui peuvent être utilisées sur des machines modestes. Ces logiciels peuvent aussi être utilisés pour valider les données avant de les intégrer dans la version, dite version intégrée, qui seule fait l'objet de ce manuel (voir La Lettre no 4 pour des informations méthodologiques complémentaires). Les modules "maître" qui permettent de réaliser des jeux de données, constituent troisième type de programmes. La version modulaire et la version maître font l'objet de descriptions séparées. Un guide du réalisateur présente l'interrelation entre ces systèmes dans le processus de réalisation des données.

Le présent manuel commence par faire une présentation globale de toutes les parties importantes du système. Puis, il passe en revue de façon plus détaillée la structure des données utilisées, le fonctionnement général puis les différents modules. En annexe on trouve une information générale sur les hypertextes pour introduire la nomenclature spécifique et une description des différentes interactions. Un manuel de référence donne une liste plus exhaustive des domaines, databases et prédicats utilisés. L'annexe A donne un extrait du cahier des charges.

Description générale

Prof'Expert est construit, en suivant la terminologie des hypertextes (voir annexe B), par interconnexion d'unités d'information (UI). Le type de l'unité, de même que le nom du lien qui permet d'y parvenir auront un effet spécifique sur la manière dont une UI utilise ses données.

Certaines de ces UI sont des textes de théorie, d'autres sont des exercices, d'autres des questions ou encore des données (énoncés, réponses), etc. L'hypertexte principal, est appelé la carte de connaissances. Il constitue à la fois le support informatif et donne la manière de se déplacer (naviguer) dans le système. Certaines opérations s'effectuent toutefois à l'aide d'un menu classique.

A chaque instant, le système est caractérisé par une présentation de l'écran donnée de façon interne par un certain nombre de faits (contextes, prédicats systèmes, touches actives, ...) que chaque UI peut modifier. L'action d'une touche (ou de la souris) est analysée par la boucle de saisie, le résultat est mis en présence des contextes. Des "experts" décident de l'état suivant, le système est mis à jour de façon interne et les modifications de l'écran correspondantes sont effectuées.

Voici par exemple les actions successives qui, d'unité d'information en unité d'information, mènent à un exercice (figure 1).

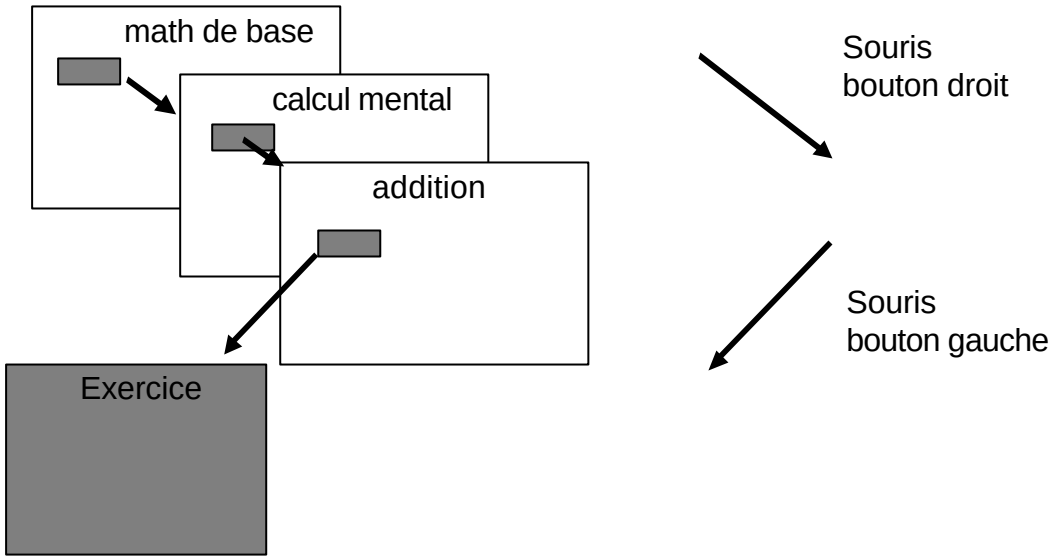


Fig 1: Choix d'un exercice

Structure générale

Chaque logiciel Prof'Expert est un projet constitué de plusieurs modules. Certains modules sont généraux à l'ensemble des logiciels, d'autres sont spécifiques (mathématique, français, ...). La différence d'un logiciel à l'autre réside dans les interactions mises à disposition et les experts actifs. Du point de vue source, chaque module correspond "physiquement" à un répertoire. Ceux-ci sont les suivants:

GENERAL outils généraux
GENMASS générateur de message
INTER interface utilisateur dont INTER/MOUSE
INTERAC interactions
HYPER hypertexte
EXPERT fonctionnement général des experts
EXPERT.FRA experts en français
EXPERT.MAT experts en mathématique
SYSTEM fonctions liées au fonctionnement général

Le détail des répertoires et fichiers par module est décrit dans le manuel de référence.

Ces modules sont indépendants et communiquent par l'intermédiaire de messages et réagissent selon le contexte matérialisé par des objets globaux contenus dans les faits de prédicat 'contexte'. Certains de ces contextes concernent le contrôle du système, les experts, etc. Le détail des contextes figure dans le manuel de référence.

L'hypertexte

L'annexe B donne les définitions générales d'un système hypertexte. Prof'Expert a été conçu sur cette base. L'hypertexte sert à la fois d'élément organisateur de la matière (carte de connaissances) et apporte des informations nécessaires (aide, guide, ...). Les exercices, les questions, énoncés, exemples, sont également à considérer comme des unités d'information. Leur structure est la même et leur traitement global suit la même cheminement. Les principaux types de noeuds et de leurs liens sont présentés dans la figure 2.

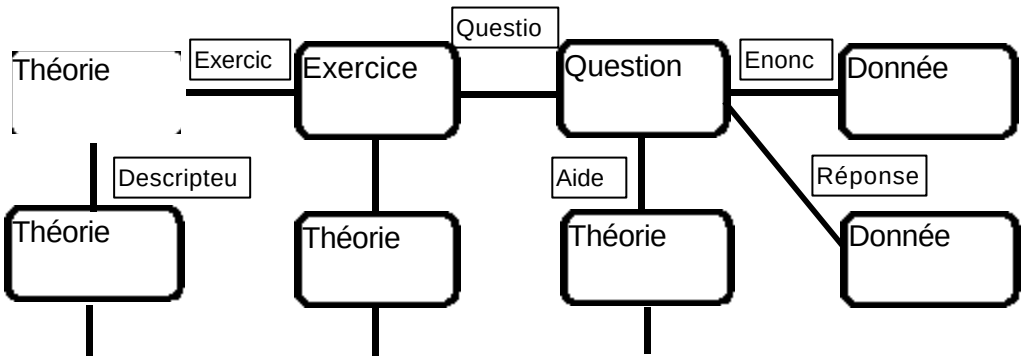


Fig 2: Types de noeuds et de liens

Le guide

Divers dispositifs peuvent modifier la configuration de l'hypertexte (noeuds accessibles, parcours possibles, etc.). Le guide est un dispositif qui établit des parcours obligés dans l'hypertexte. Ce dispositif intervient en situation d'aide, il conduit l'élève vers la solution ou lui indique la démarche à suivre.

Les variables dans l'hypertexte

Le contenu d'un texte de l'hypertexte peut contenir des joker (%). Dans ce cas, des attributs "variable" contiennent le nom des variables à insérer (en découpant l'espace nécessaire si il y a suffisamment d'espace, en décalant sinon). L'insertion se fait avant affichage. Le système va chercher la valeur des variables dans le prédicat *valeurVar(nom,valeur)*. Si cette valeur n'existe pas, c'est le nom même de la variable qui est inséré. Ces variables proviennent des données. Il se peut également que la variable soit un hyperchamp. Dans ce cas, au moment de l'évaluation d'un nouveau fait: *handle(nom,handle)* est créé et est utilisé pour mettre la valeur en hyperchamp. Si % se trouve en hyperchamp, le même phénomène se passe. Mais dans ce cas le handle est fixe. Dans l'autre cas, il peut être variable.

L'héritage dans les « link »

Une unité d'information hérite des link des unités qui l'ont précédée et qui n'ont pas encore été refermées. Cela signifie qu'une question "hérite" des aides et des exemples qui ont été définis dans l'exercice, voir de ceux d'une unité d'information antérieure.

Les contextes

Les contextes sont des faits, dans le langage des systèmes à base de connaissances, qui modèlent l'interaction. Type de l'interaction, variante, experts à disposition, tempo, tolérance, feed-back, reprise, suivi, etc. sont tous des éléments donnés par des contextes particuliers. Les contextes, sous la forme d'une chaîne listée, constituent un attribut particulier d'un noeud. Chaque unité d'information peut être pourvue d'un tel attribut 'contexte'.

Lorsqu'une unité est activée, les nouveaux contextes remplacent les précédents selon certaines lois liées au privilège de chacun. Lorsqu'une unité d'information est désactivée, les contextes antérieurs sont rétablis, parfois en gardant les valeurs qu'un autre environnement peut leur avoir attribué.

Un contexte activé est un fait dont la structure est la suivante: contexte(CONTEXTE, PRIVILEGE, ETAT)

Les exercices

Un 'exercice' constitue un pôle du fonctionnement du système. Un exercice est un noeud particulier qui supporte la plupart des attributs liés à une interaction. Un exercice est toujours liés à une UI de la carte des connaissances. Des questions lui sont liées. Aux questions sont liées des données servant soit d'énoncé, soit de réponse. Un exercice se construit à l'aide de trois composants: les données, les experts et l'interaction. Mais chaque 'question' d'un même exercice peut aussi posséder sa propre interaction, auquel cas là, elle joue le même rôle qu'un exercice.

Les données

Les données sont rassemblées dans des fichiers (hyperbases) dont l'élément fondamental, l'enregistrement ou le noeud est une structure permettant d'enregistrer une information et caractérisée par des attributs en nombre indéterminé. Cette notion a été reprise, de même que le principe de la Hypertext Abstract Machine (HAM) chez PDC-Prolog.

Une partie des éléments servant à un moment donné sont chargés dans une database globale particulière (données).

Les experts

Les "experts" sont constitués de contextes particuliers qui dirigent des règles permettant de structurer les données, corriger les réponses, donner les évaluations.

Les interactions

Une interaction est définie par un contexte principal et quelques contextes accessoires. Elle est l'environnement global dans lequel se déroule un exercice: position de l'énoncé, type de manière de donner la réponse, aides àdisposition, etc.

Les différents types d'exercices et les interactions sont décrits dans l'annexe C.

Evaluation et suivi

Evaluation d'une question

A la fin de chaque question, un résultat qui dépend du type d'exercice, est enregistré: réponse juste au fausse, performance en temps, degré d'exactitude, nombre d'essais, etc.

Evaluation en fin d'exercice

En fin d'exercice, les résultats aux différentes questions sont réunis dans un indice de performance global. Le nombre d'appels àl'aide est comptabilisé, etc. Lorsque l'option 'suivi' est active (ce qui demande àl'utilisateur de s'inscrire àl'aide d'un sobriquet) cette information est enregistrée. L'utilisateur peut donc garder une trace de ses résultats. Dans la forme actuelle, ce suivi est constitué des exercices réalisés (nom de l'exercice) de l'heure du travail, de l'état à la fin de l'exercice (arrêt ou fin de l'exercice) et d'une performance globale sur les questions travaillées. Les résultats sont ventilés par notions travaillées. D'autres actions peuvent être enregistrées àfin de recherche. Dans une version ultérieure, le suivi sera enregistré dans une hyperbase, pendant de la carte des connaissances, qui pourra également être parcourue. Elle pourra diriger des apprentissages guidés et personnalisés.

Reprise et erreurs

Il est possible d'arrêter un exercice en cours de travail. Tant qu'un nouvel exercice n'est pas initialisé, l'exercice non terminé peut être repris ou sauvegarder pour une reprise ultérieure. Selon le même schéma, il est également possible de réinsérer toutes les questions non réussies dans un exercice et de consulter les énoncés de toutes les

questions où une erreur a été commise. Les exercices terminés sont annotés. Ainsi, il est toujours possible de savoir si un exercice a déjà été effectué ou non. Lorsque le suivi est actif, ces annotations sont gardées d'une session à l'autre.

Les messages

Les messages constituent une part importante de l'interaction. Leur but est d'inciter à une certaine dynamique plutôt qu'à donner des explications. En effet, il semble plus adapté de faire suivre une erreur par une nouvelle question du même type que par un message explicatif. Les messages peuvent demander de rester vigilant ou apporter des éléments de diversion. Certains messages sont communs à l'ensemble des exercices. D'autres peuvent être liés à un exercice ou une question particulière.

Structure des données

Les hyperbases

Les données sont enregistrées selon un format très souple. Chaque donnée est un noeud hypertexte, c'est-à-dire, selon le modèle HAM, une structure de type:

node(BASE,REF,ATTLIST,CONTENT,DATE,VERSION,LINKLIST)

BASE et REF concerne la localisation physique du noeud. ATTLIST est la liste des attributs du noeud: nom (name), printname, type, etc. CONTENT est le contenu informatif liés au noeud. DATE et VERSION concernent la création et la mise à jour du noeud. Quant à LINKLIST, il s'agit de la liste des liens.

Chaque lien est de la forme:

link(HANDLE,ATTLIST,BASE,REF,VERSION).

Le HANDLE est le concept concerné par le lien (le descripteur), ATTLIST la liste des attributs (principalement le nom), BASE et REF déterminent l'UI cible.

Chaque élément de donnée est un "objet" dont la liste est énumérée dans l'annexe D.

Les données internes

Il y a plusieurs données regroupées dans des databases internes:

La database globale contient les contextes (contexte(TypeContexte,Privilège,oui_non)), les faits systèmes (système (system)), les stacks (stackOfContext(stackContextList),

currentLL(STRING,STRING,linklist)) de même que quelques autres informations et flags.

La database globale - données - contient tous les éléments nécessaire àla réalisation d'exercice. Cette database est constituée des éléments contenus dans les hyperbases: deux types de prédicats permettent de conserver plusieurs types d'informations: donnée (donnéeGlob, param), donnéeStr (donnéeStrGlob, STRING)). D'autres sont plus spécifiques (valeurVar (STRING, DON), distractlist (stringlist), currQues(link), currExer(link), etc.

La database globale - actionDb - collectionne les actions (actionStack (SYMBOL)). Un démon (lié àassert et retract) modifie la ligne de status.

La database globale - userInfo - garde des informations sur les utilisateurs. Actuellement *user(SYMBOL,STRING,STRING,STRING)* permet de garder le sobriquet, le fichier de suivi, le fichier utilisé pour les sauvegardes et d'autres informations sur l'utilisateur (nom).

Fonctionnement général

Le système Prof'Expert fonctionne sur la base de deux cycles: une boucle de saisie qui permet les réactions du système aux actions extérieures et un moteur hypertexte qui gère les enchaînements dirigés par les données. Un exercice (ou une question) est aussi à la source d'un processus d'une certaine importance en partie standard (un exercice est une unité d'information particulière) mais avec un nombre de spécificités important.

Boucle de saisie

La boucle de saisie est basée sur un prédicat ‘repeat’ standard. A chaque cycle, le système détecte une action et l'exécute (prédicat: *envoie(Action,Key)*).

La détection d'une action se fait selon l'ordre de priorité suivant:

-action stockée dans une file d'attente,
-action par défaut étant donné le cycle de saisie actif,
-action déterminée en fonction du cycle de saisie actif et de la touche activée en situation standard
-action déterminée en fonction du cycle de saisie actif et de la touche activée en situation d'éditeur.

Moteur hypertexte

Le fonctionnement du moteur hypertexte est lié à deux paramètres: le concept évoqué (handle) et le nom du lien appelant. Les liens peuvent être sélectionnés en fonction de ces deux paramètres et d'autres attributs encore. Par exemple, il est possible de rendre certains liens inactifs (en fonction du niveau de l'apprenant ou tant qu'un noeud donné n'aura pas été visité). Le moteur agit en trois étapes:

-étape d'initialisation: il s'agit de la recherche du node, de la mise en stack de la 'linklist' (pour l'héritage) et de la mise en mémoire de la liste des liens qui répondent aux paramètres d'appel. Cette étape appelle la deuxième en ajoutant aux paramètres le premier link.

-étape de modification des contextes: cette étape qui appelle la troisième sert principalement à établir les nouveaux contextes.

-étape de l'action: une action liée au paramètres et au link est effectuée. L'action consiste en le chargement de données, la mise en route d'un exercice, etc. Les différentes actions seront détaillées ultérieurement.

Gestion générale des exercices

Le cas des exercices fournit à la fois un exemple du fonctionnement d'un cycle du moteur et un cas particulier par l'ampleur des actions qui sont associées à ce cycle. On distingue les étapes suivantes en considérant tous les appels au moteur dès le chargement d'un exercice.

1. *Cycle sur un exercice, modification des contextes*

    Mémoriser les anciens contextes.
    Etablir les nouveaux contextes.
    Mémoriser ou utiliser d'autres attributs (name, printname, consigne, etc.).
    Lancer le moteur sur le handle question.

2. *Niveau question, début du cycle*

Lors de ce nouvel appel, le moteur commence par mettre en stack la linklist précédente (il y aura peut-être lieu de se référer à l'aide stockée à ce niveau). Elle établit la liste courante (dans l'ordre original ou dans un ordre aléatoire selon le contexte) des 'link' à suivre, chacun pointant sur une question.

3. Niveau question, suite du cycle (avec répétition)

    Prise en compte du premier link, recherche du noeud lié.
    Modification des contextes.
    Traiter la question en mémorisant les éléments importants et en lançant le moteur pour:
        l'énoncé,
        la réponse (selon le contexte: la réponse est cherchée dans database ou calculée selon expert didactique).
    Déclenchement de l'interaction.

Valider la réponse avec ou sans effacement de la question.
Rétablir les contextes et le stack des ‘link’ au niveau "exercice".

Mécanisme des contextes

De façon générale, c'est le "contexte" qui détermine la manière d'agir des différents éléments (interaction, permission ou interdiction de certaines fonctions, experts - indication des tolérances...). Ce contexte est matérialisé par un ensemble de faits, qui dirigent le fonctionnement du système. Il y en a de plusieurs types et leur énumération se trouve dans le manuel de référence de même que des précisions techniques sur leur traitement informatique. Les contextes sont établis par le créateur des bases d'exercices. Certains concernent le contrôle général de l'interaction (reprise, ...) ou encore la pédagogie générale utilisée (suivi, ...). D'autres sont liés plus spécifiquement àdes exercice (experts, tempo, ...).
Ils ont la particularité de pouvoir être modifiés via le menu ‘options’ par l'utilisateur, par le moteur hypertexte selon les données de l'exercice ou encore par les experts. Certains contextes peuvent être fixés au départ dans un fichier de configuration (pe.cfg) selon le même format que dans la base des exercice. Un autre manière de fixer des contextes de départ est de les insérer dans un noeud de la Carte des connaissances (le noeud sur lequel pointe ui_entrée, par exemple).

Chaque contexte est accompagné de son privilège. Ceux-ci sont:

-aucun: le contexte accompagné de ce privilège n'est plus modifié. Par exemple si un fichier de configuration ou une unité de la carte des connaissance établit le contexte suivi(oui) avec ce privilège, il ne sera plus modifié même si un exercice porte ce contexte avec une nouvelle valeur. Cela signifie que tous les utilisateurs du système devront s'identifier et verront leurs résultats enregistrés, quelques soient les options ultérieures qui pourraient avoir été prévues par les réalisateurs

-système: le contexte ayant ce privilège peut être modifié par le système. Dans le cas de l'exemple précédent, le contexte établit par la CC peut être modifié par un exercice.

-tout: le contexte peut être modifié par l'utilisateur grâce au menu ‘options’. Les différentes valeurs que peut prendre le contexte sont établies par le créateur. La première valeur donnée correspondra au contexte actif. Les valeurs décidées par l'utilisateurs restent actives tant que le contexte est prévu par l'environnement général. Dans le cas contraire, le contexte est retiré. Il pourra réapparaître ultérieurement mais avec la valeur par défaut. La valeur introduite par l'utilisateur sera oubliée.

Le mécanisme des contextes procède par une technique d'héritage double; en descente et en remontée. Chaque noeud activé ajoute ses contextes propres à ceux existant (descente), ceci en modifiant les valeurs si les privilèges le permettent. En quittant un noeud (remontée), les contextes initiaux sont rétablis. Au cas où certaines valeurs ont été modifiées au niveau n+1 (par le système ou par les options), les

nouvelles valeurs remontent au niveau n. Elles sont par contre perdues si le contexte n'est pas défini àce niveau supérieur. Le contexte ‘suivi’ fait exception àla règle, dans la mesure où, le privilège le permettant, les valeurs modifiées sont conservées à la remontée, même si ce contexte n'était pas préalablement défini.

Voici quelques contextes communs àtoutes les configurations du système:

Suivi: suivi(oui|non),
Controle: ordre("aléatoire"|"original"), remise(avec|sans),
Objectif (Elève): objectif(symbol), niveau(integer ou symbol),
Ton: ton(grossier| poli| neutre| fantaisiste| impertinent).

Ils sont établis en général dès le fichier de configuration, mais peuvent aussi apparaître au menu et être modifiés par les experts!

Quelques questions sont pendantes. Celle d'un contexte concernant la possibilité ou non d'utiliser l'aide ponctuelle. A voir également pour un contexte priorité, qui existe dans la version modulaire, mais dont l'intégration dans la version intégrée sera liée au suivi global.

Les autres contextes sont cités par interaction dans le manuel de référence.

Questions, données, experts et interaction

Un exercice (ou une question) est construit par les trois éléments distincts mais difficilement séparables que sont les données, les experts et l'interaction. On peut encore évoquer les contextes dans la mesure où ils commandent les trois éléments ensemble. Les données sont chargées en premier par le moteur hypertexte àl'aide de son fonctionnement naturel. Ensuite, un dialogue permanent entre interaction et expert produira le fonctionnement de l'exercice qui testera l'utilisateur. La forme concrète que prendra ce jeu entre interaction et experts est entièrement déterminée par les contextes.
Afin de rendre le contrôle àla boucle de saisie au moment adéquat, l'interaction a été découpée en plusieurs étapes. Dans son fonctionnement le plus simple elle a la structure suivante:
-début: Il s’agit de la construction de l'environnement de la question, principalement les différents éléments de l'écran, la mise en place du (des) cycle(s) de saisie (en règle générale, chaque type d'interaction àson propre fichier de définition de clés) avant la première saisie.
-évalinterm: Cette partie est activée par la détection d'un événement indiquant qu'une réponse a été saisie. Un appel àl'expert qui procède àl'évaluation de cette réponse est fait.
-fin: Démontage de l'environnement mis en place par ‘début’ et appel à*mot_hypReturn*. Appelé par *postsaisie(interaction)* ou par l'action interactionfin (plus utilisé?).

Durant la phase de saisie plusieurs événements liés à l'interaction peuvent être déclenchés. Il s'agit principalement de:
-passer: pour ne pas répondre àune question,
-revoir: suivant le cas, affiche l'énoncé (flash), ou une réponse possible,
-suite: après une interruption,
-ralentir: pour les interactions travaillant la vitesse,
-guide: un guide mène àla solution.

Tout ces parties sont en principe déclenchée par des touches décrites dans "*.key" ou un appel au menu. La saisie d'une réponse peut aussi être interrompue par un appel à la modification d'une option ou un appel àl'aide qui ne font pas partie de l'interaction proprement dite.

Il existe encore:
-réponse: un cas spécial pour l'interaction flash, ou parfois chaque caractère est évalué,
-annule: idem pour flash, lors d'une interruption non évaluée.

Interaction et cycle de saisie

En principe, l'interaction a deux cycles de saisie. Le cycle *interaction* qui est recouvert lors de la saisie par le cycle de saisie de la méthode de saisie propre àl'interaction particulière (souvent *lec* de l'éditeur). Ce deuxième cycle est souvent annulé lors d'une intervention de type aide, revoir etc., et l'appel àl'action proprement dite (recherche et affichage de l'aide, par exemple) définie au niveau du cycle *interaction*. Ici, selon les contextes, on lie un fichier de clés au symbole interaction (voir boucle de saisie).

Les experts

Les experts, comme contextes, prolongent les caractéristiques données par le type et le mode d'interaction, sont utilisables en tout temps, mais principalement lors de la mise en forme d'un exercice et lors de l'analyse des réponses. Il y a quatre contextes dirigeant les expertises; experts(dom()), experts(did()), experts(psy()), experts(mode()). L'expert du domaine (dom) gère les règles fondamentales. Dans le cas des mathématiques se sont les règles d'analyse des expressions. L'expert didactique gère la manière d'évaluer les réponses. Pour un même type d'interaction on peut s'intéresser àdifférentes évaluations. Dans le cas de *calcul* on s'intéresse àla justesse du résultat, alors que dans le cas de *estime* on tient compte de la rapidité de la réponse et du degré de précision atteint.

Toutefois, ces experts ne sont pas des procédures séparées! Il n'y a qu'un seul appel: le prédicat callExpert(_,_). Le fonctionnement des règles dépend principalement des quatre type de faits donnés par les contextes. Ce cas des experts montrent bien le principe de fonctionnement du système: créer des environnements qui conditionnent son comportement. Le choix des experts importe moins que le nombre de possibilités qu'ils offrent pour représenter tous les cas! C'est une caractéristique des systèmes à base de connaissances. C'est un paquet de règles qui représente un domaine ou

'objet'. Il n'y a pas, comme dans le schéma classique des ITS, la localisation d'experts particuliers.

Les différentes fonctions des experts, représentées par leur paramètres sont:

-corrige: vérification que les touches activées sont autorisées (cas des flash),
-initEvaluation: préparation de la structure pour recevoir les informations liées à l'analyse des erreurs et comportement de l'apprenant,
-établitRAct: mise en forme de la rétroaction,
-établitEnoncé: mise en forme de l'énoncé d'une question àpartir de données fournies,
-fabriqueEnoncé: fabrication automatique d'un énoncé,
-majénoncé: mise à jour de l'énoncé lorsque celui-ci évolue en fonction des réponses données,
-structure: structuration de la réponse donnée (passage d'une chaîne de caractère àune structure),
-ajTouchesSpécifiques: choix d'un jeu de touches actives avec leur spécification de fonction,
-établitParamètres: calcul des divers paramètres nécessaire à la présentation de l'énoncé. Etablissement de la réponse si nécessaire et des méthodes pour y parvenir,
-paramSup: complément pour le choix des distracteurs, par exemple,
-évaluationInterm: gère l'ensemble de l'analyse d'une réponse,
-évaluation: juge la réponse donnée,
-évaluationFinale: compile l'ensemble des résultats aux différentes questions pour en fait un résumé,
-formater: utilitaire de formatage (plus utilisé),
-actionDidactique: modification de l'interaction en cours de travail.

Les experts font appel àdiverses procédures. En particulier:

-recherche du participe passé d'un infinitif (et ultérieurement: conjugaison, pluriel, féminin),
-parser pour structurer une expression algébrique (ultérieurement: booléenne). La forme internes des expressions est décrite dans le manuel de référence,
-comparateurs divers: en particulier d'expressions algébriques littérales,
-recherche de méthodes pour calculer une 'quatrième proportionnelle',
-recherche de procédures pour réaliser un calcul arithmétique ou la résolution d'un système d'équations linéaires.

Evaluation et suivi

Evaluation de la question

Une database - expertise - mémorise les valeurs significatives résultant de l'évaluation de chaque question. Les éléments suivants sont mémorisés sous formes de listes dans le prédicat evaluation: nom (string), concept (string), tempsTrav (integer), précision

(real), rapidité (real), essai (integer), nRemplacements(integer), repJuste (oui_non), revoir (integer), theorie (integer), aide(string), taux(integer), abandon.

Les experts (dom, did, mode) sont des éléments qui sont également mémorisés par question (chaque question peut faire appel àune interaction particulière et des experts spécifiques). Ces informations, de même que le concept, permettent de ventiler les résultats. Pour des raisons historiques, le nom des interactions peut influencer sur les expertises ultérieures. Pour cela, lors de l'initialisation de l'évaluation les noms des experts utilisés peut différer du nom enregistré dans les données (interaction *flash* + domaine *calcul* est résumé par domaine *calculm*, par exemple).

Un autre prédicat de la même database, textEval, est utilisé pour accumulé lors de la mise en forme de l'évaluation, les textes nécessaires.

Le concept est obtenu dans un contexte particulier, lié soit àl'exercice, soit àla question. Il est établi par défaut comme étant le nom du handle qui a conduit àl'exercice!

Evaluation finale de l'exercice

L'expert didactique reprend les éléments de l'évaluation, en fait l'analyse et rend une synthèse. L'évaluation finale est activée lorsque la remontée dans le moteur hypertexte détecte que l'on quitte un noeud exercice (cf. *otherLink*)

Un option *suivi* existe qui permet à l'utilisateur d'enregistrer automatiquement les résultats obtenus. Lors de l'enclenchement de cette option, l'utilisateur est invité àdonné son sobriquet (nom d'un fichier, donc 8 caractères alphanumériques au maximum), et lors de la première utilisation de s'inscrire en donnant son nom. Le suivi peut être rendu obligatoire en inscrivant le contexte correspondant sans aucun privilège.

Système de reprise et consultation des erreurs

Lorsqu'un exercice est arrêté, de même qu'à la fin d'un exercice, les données de l'exercice sont sauvegardées dans un fichier temporaire, initialisé lors du début de chaque nouvel exercice. La commande du menu 'Sauvegarder un exercice en cours' copie ce fichier dans un fichier dont le nom est donné par l'utilisateur (ou de nom <sobriquet>.EXR lorsque le suivi est actif), ce qui permet un rechargement et une reprise ultérieure. Tant qu'un nouvel exercice n'est pas initialisé, il est possible de reprendre un exercice, de consulter la liste des énoncés des questions où une erreur a été commise ou de retravailler ces mêmes questions. C'est cette dernière possibilité qui justifie la sauvegarde des données en fin d'exercice. La reprise des erreurs en cours d'exercices provoque la remise des questions avec erreur dans la liste des questions àtraiter.

Les options reprises s'organisent de la façon suivante:

En cours d'exercices
Reprise des erreurs: les questions avec erreur sont remises dans la liste courante des questions àtraiter.

Après un arrêt
Reprise: reprend les questions non traitées.
Reprise des erreurs: les questions non traitées et les questions avec erreur sont reprises.

Après une fin normale
Reprise: désactivée.
Reprise des erreurs: les questions avec erreur sont reprises.

Un exercice terminé est enregistré dans le prédicat db_trace(name,printname,handle) de la database trace (interfacé par le prédicat global: reg_exer). Cette base est sauvée lorsque le suivi est actif.

Système des messages

La création de tous les messages (status, messages d'erreurs, aidetype , ...) est assurée par un seul prédicat *constMSG (msglist,STRING),* où msglist est une liste de mots-clés (voir manuel de référence). Le message est construit ou choisi àpartir de ces mots-clés et du contexte ton et d'autres éléments tel que le nombre de fois où un type de message a déjàété utilisé.

Les messages sont affichés selon leur style: correspondant au domaines: *msgSty = evFin ; dlg ; rAct ; hMenu* A l'aide du prédicat *message(msgSty,String)*.

Description des modules

Objet moteur hypertexte

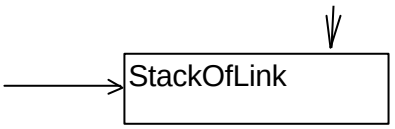
Le centre du logiciel est le moteur hypertexte (figure 3), chargé de gérer le chargement des informations contenues dans les hyperbases. Le prédicat accessible depuis l'extérieur, donc global, est *init_moteur_hyp (Lien, Handle).*

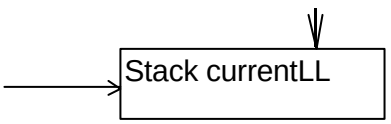
Handle est la référence au lien(s) àtraiter du noeud courant. Le Handle représente un concept. Selon le niveau, l'idée de concept change. Dans la carte des connaissances, un concept est une notion (objectif) du domaine. En cours d'exercice, les concepts sont les différents types d'aide! *Lien* est une contrainte supplémentaire (nom du lien) dans le cas où il existe plusieurs liens possédant un même ‘handle’. Par exemple, un exercice et un descripteur peuvent être liés au même concept.

Actuellement le moteur hypertexte est appelé à partir de la rubrique ‘choix’ du menu ‘exercice’. Ce premier appel a la forme init_moteur_hyp ("choix","init"). Il ouvre la database des objectifs, puis il charge "manuellement" le noeud qui porte le nom "ui_entrée" (appelé ainsi, car son link fait référence au premier noeud à traiter de la base). Ensuite un appel standard est fait à init_moteur_hyp avec la contrainte "descripteur", « ui_entrée ».

Mis à part le cas spécial du handle « ui_entrée », *init_moteur_hyp* crée une sous-liste de liens répondant aux exigences de Lien et Handle (*get_link)*. Une sélection plus fine (choix par menu, choix aléatoire, autre) à lieu dans (*select_link*). Les éléments sélectionnés sont enregistrés dans *currentLL*. Après chaque traitement, cette liste est remise à jour, et l'élément suivant de la liste exécuté par *otherlink*. La fin du traitement est indiqué par *mot_hypReturn.* Dans le cas de la sortie d'un lien exercice, *otherLink* commande le départ de l'évaluation finale.

moteur_hyp, pour sa part, charge le noeud correspondant au lien traité (*nextNode*), gère les contextes (voir héritage des contextes), effectue une action correspondant à Handle et Lien (*action*). Notons qu’un noeud objectif qui n'est pas rechargé s'il existe déjà dans une instance précédente (*existNode*). Dans ce cas, tous les noeuds intermédiaires sont automatiquement refermés. Par ailleurs, en cours d’exercice, tous les liens menant aux exercices sont désactivés. Ces deux particularités du moteur ont été introduites pour diminuer les risques que les utilisateurs se perdent dans l’hypertexte.





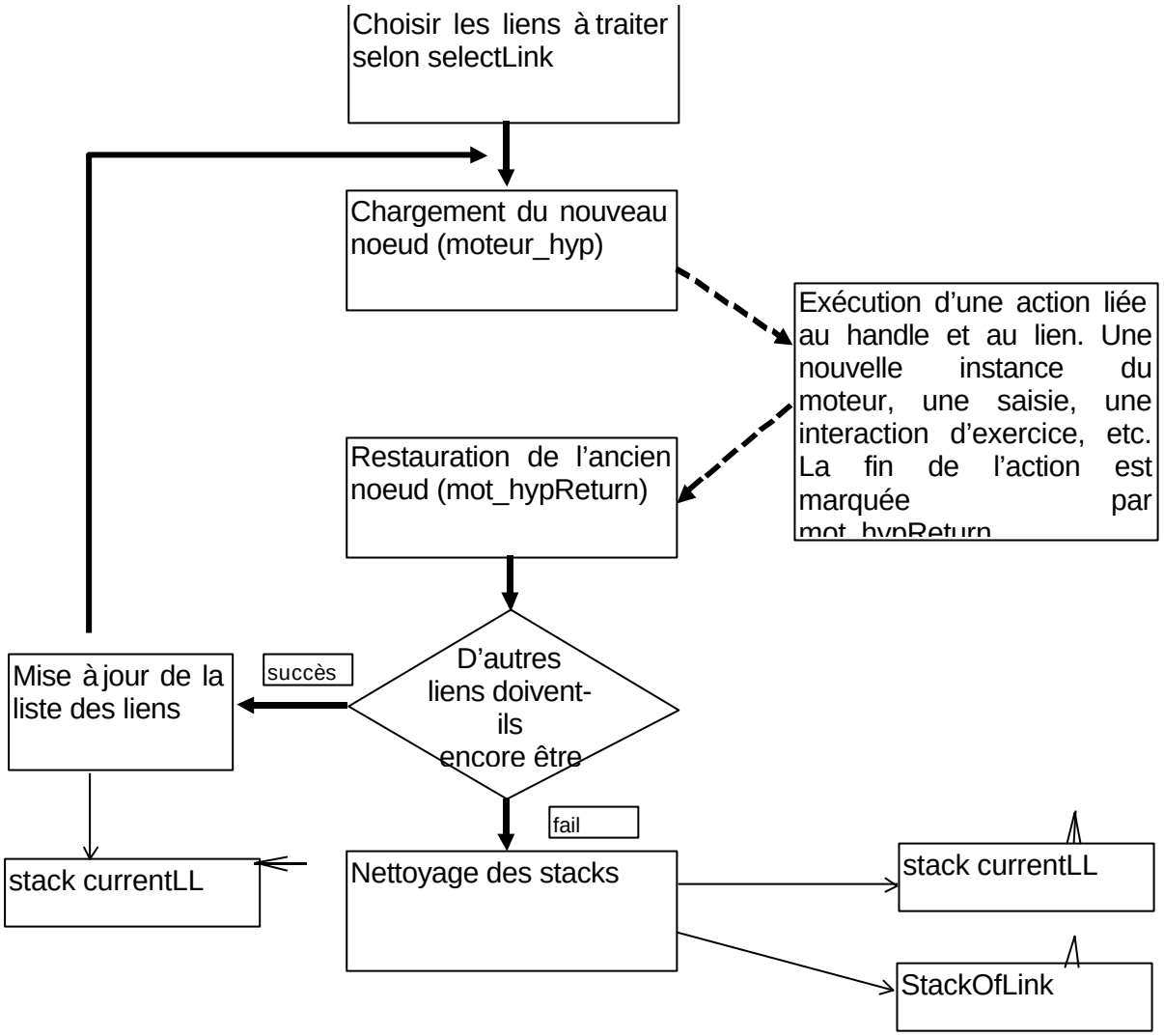


Fig 3: le cycle du moteur hypertexte

action est le prédicat qui exécute les fonctions liées au noeud nouvellement chargé. Il peut s'agir du chargement d'un exercice, d'une question, d'un énoncé, d'une réponse, etc. Cette action est conditionnée par la double contrainte du Handle et du Lien et peut contenir un nouvel appel àinit_moteur_hyp. Dans ce cas le nouveau cycle du moteur est complètement exécuté avant de revenir àce niveau.

Dans le cas du lien "descripteur", action effectue le chargement d'un (nouvel) éditeur, contenant un texte et des hyperchamps. L'activation d'un de ses hyperchamps provoquera un nouvel appel de init_moteur_hyp.

Héritage dans les liens

Pour tous les types de lien (descripteurs, données complémentaires, aide, distracteur, etc.), la recherche des liens se fait en deux étapes: d'abord dans la liste des liens du noeud courant, puis, en cas d'échec, on tente de trouver un lien adéquat dans le stack des listes de liens des instances précédantes (*stackOfLink* de la database -hyper). Ce dispositif est extrêmement puissant. En particulier, pour assurer des « défauts », il suffit d'implanter des liens au niveau de la première UI d'une carte de connaissances. Par

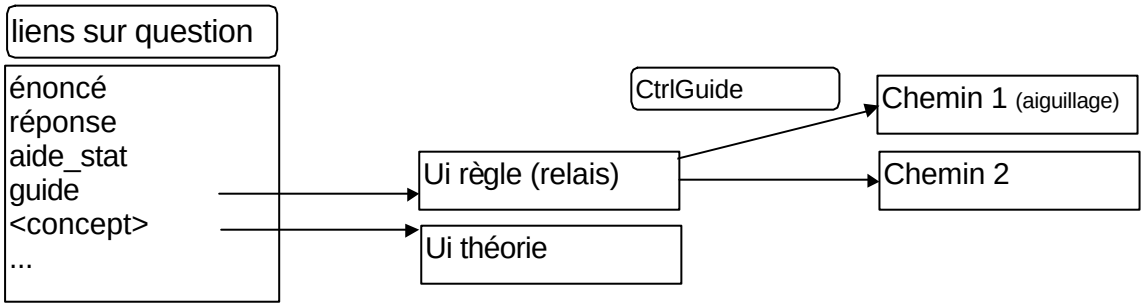
contre, il est difficile de, par exemple, supprimer l'aide (ce que certains souhaiteraient afin de créer des situations de test). En effet, par héritage et défauts multiples le système trouve toujours un moyen d'informer l'utilisateur. L'absence de lien devrait être marquée de façon explicite.

Attributs complémentaires

Les liens peuvent encore être sélectionnés en fonction d'attributs complémentaire. En particulier un lien de nom descripteur est inhibé par l'attribut att(exercice,HH) tant que l'exercice lié au handle HH n'a pas été terminé. Si HH est la chaîne vide, l'exercice considéré sera celui possèdant même handle que le descripteur.

Le guide

Le dispositif du guide est un exemple de la possibilité d'agir sur les parcours possibles. Ce dispositif est implanté de la manière suivante: tout d'abord la carte des connaissances contient des nodes de type relais avec les noms des règles (= nom du node). Des 'handle' de nom CtrlGuide fournissent les guides proprement dits. Le guide est une chaîne listée (chaîne avec des séparateurs |) apparaissant comme valeur de l'attribut 'chemin' dans un node de type 'aiguillage'. Le nom du node est le nom de la règle, suivi d'un indice (il peut y avoir plusieurs chemins possibles pour une règle).



La question ou l'exercice demandant le guide contient un lien de handle 'guide' pointant sur la règle. De plus il contient un lien de handle le premier concept du chemin pointant sur la première UI d'aide (qui se trouve souvent être celle de l'aide statique).

Ce chemin est implanté sous la forme d'une database ctrlGuide(NOM,CHEMIN). Dans certaines conditions (en mode guide) le système (via le prédicat selectLink) vérifie que le handle sélectionné est le premier d'une liste avec empilement et désempilement. Attention, il peut y avoir plusieurs chemins possibles.

Les bases supplémentaires sont ouvertes au moment de leur besoin, c'est-à-dire au moment où elles apparaissent dans les informations du lien. Concrètement, au moment de changer de node (*nextNode*), le prédicat *isFileOpen* contrôle l'existence de la base, et l'ouvre le cas échéant.

L'information concernant les databases est contenue dans le prédicat *baseSelect.* Il faut rappeler à ce propos que les informations du lien contiennent le numéro d'enregistrement du noeud, mais aussi le nom de la base (nom du fichier). Mais un domaine db_selector défini les noms logiques des hyperbases (actuellement les noms logiques disponibles sont hyperbase1, hyperbase2, hyperbase3, hyperbase4, hyperbase5, hyperbase6, hyperbase7).

Lors de la première ouverture d'une hyperbase, le prédicat *baseSelect* associe le nom physique, sans path!, au sélecteur hyperbase1 et une chaîne vide aux autres. Lors de l'ouverture d'une base supplémentaire le nom physique, toujours sans path, de la nouvelle base est attribué au premier sélecteur disponible. Lors de la fermeture d'une hyperbase, son sélecteur est libéré en étant à nouveau associé à une chaîne vide. Autrement dit, tout accès à une hyperbase se fait par la consultation de *baseSelect* à partir du nom physique de la base.

Le problème de la fermeture des bases est délicat. Comment savoir si une hyperbase n'est plus utilisée (les environnement font appels au db_selecteur et non au nom des fichiers !).

En ce qui concerne l'établissement des informations effectives, celles-ci peuvent être: entièrement déterminées, soit calculées (cas de certains énoncés), soit encore complétées par la valeur de variable. Ce point a déjà été évoqué, il est précisé ici. Pour compléter une UI, le système va chercher la valeur des variables dans le prédicat *valeurVar(nom,valeur)*. Si cette valeur n'existe pas, c'est le nom même de la variable qui est inséré. Ces variables proviennent des données. L'annexe E décrit quelques-unes des variables utilisées. En ce qui concerne les hyperchamps variables, trois techniques existent:

1) La variable, dans les données est associée à un handle. Dans ce cas le handle (suite à une sauvegarde dans le fait *handle(nom,handle)*) passe à la valeur de la variable. Cette méthode est utilisée pour un lien fixe, indépendamment de la valeur de la variable. Mais le handle peut varier d'un exercice à l'autre.

2) Le joker % se trouve en hyperchamp, le même phénomène se passe. Mais dans ce cas le handle est fixe.

3) La valeur de la variable contient le handle. Cette méthode permet d'avoir des handle "calculés". Cette technique est utilisée au niveau du code du programme, les deux autres apparaissent dans la réalisation de données.

Objet Interface Utilisateur

Boucle de saisie (Clavier ou Souris)

La boucle de saisie est la boucle principale du système. Sauf dans des cas particuliers bien précis, chaque fois que le système attend une instruction saisie au clavier, il remonte au niveau de cette première boucle.

Le schéma suivant montre le principe général de la boucle, que l'on pourrait aussi appeler boucle de la gestion des événements. L'analyse de l'événement dépend du cycle de saisie courant. C'est le cycle de saisie qui détermine quels sont les événements dont le système tient compte, quels actions y sont liées et l'utilisation ou non de l'éditeur prolog. Pour plus de détails sur la notion de cycle de saisie, voir plus bas.

Le prédicat *cycle_de_saisie* sert à la reconnaissance des contextes et "distribue" les tâches à des prédicats propre à l'interaction et au fonctionnement propre à ces contextes.

A. *Attente d'un événement*

Lorsque le système se trouve en attente d'un événement, il procède de la manière suivante:

1. Tient compte d'un événement placé en attente par l'action précédente. La commande et la clef sont stockées dans la database memActionDb. De telles actions sont placées par les prédicats *memKey* et *memAction* (le second est préférable).
2. Tient compte d'une action par défaut pour un certain cycle de saisie. Cette action par défaut débouche, en principe, sur un nouveau cycle de saisie de peur de provoquer une boucle sans fin.
3. Attend une action de clavier ou de souris.

La lecture du clavier ou de la souris se base sur l'outil PDC Prolog Readkey qui est une boucle qui contrôle le gestionnaire de souris tant qu'une touche de la souris ou une touche du clavier n'est pas appuyé. Si une touche du clavier est activée. Readkey transforme les informations numériques récupérés par prolog en symboles. La liste des symboles des touches est donnée dans le manuel de référence.

Readkey ne teste que le clic gauche ou droit de la souris. Les fonctions "tiré" ne sont pas supportées!

B. *Analyse de l'événement*

Pour les points A.1 et A.2, l'événement est connu au moment de sa détection. Lorsque, par contre, le clavier ou la souris sont activés, il faut une analyse supplémentaire pour déterminer si une action doit suivre l'événement détecté.

Cette adéquation entre une touche ou un click et une action possible est déterminée par le prédicat *analyseKey*. Il compare événement clavier-souris à la liste des clés actives (*activeKey*). La comparaison se fait en deux temps:

1. Est-ce que la touche particulière activée correspond àune action?
2. Si non, est-ce qu'elle appartient àun groupe de clés auquel correspond une action. A cette fin le prédicat *grKey* détermine àquels groupes la touche appuyée appartient. Une touche peut appartenir àplusieurs groupes. Si aucune action ne correspond au premier groupe, d'autres groupes possibles sont cherchés. Pour la définition des groupes de clés et l'ordre dont analyseKey en tient compte, voir le manuel de référence.

Le prédicat *ActiveKey* permet de définir une touche alternative dont le système tiendra compte par la suite plutôt que de la touche activée. Cela permet de régler les problèmes de touches équivalentes au niveau de la saisie! Le prédicat *choseKey* choisit la clé alternative si elle est différente de *nil*.

C*. L'éditeur prolog et la boucle de saisie*

Pour la gestion de l'hypertexte, il est agréable d'utiliser l'éditeur livré par prolog. Cet éditeur réagit cependant à des constantes numériques plutôt qu'à des symboles et possède son propre *readkey* et système de transformation. Il n'a pas été possible de n'utiliser que la lecture de l'éditeur car celui-ci ne gère pas la souris. D'autre part, les symboles permettent plus de souplesse que les constantes. La solution choisie a été de replacer le caractère analysé dans le « buffer » du clavier lorsqu'il est destiné à l'éditeur. L'utilisation de l'éditeur au niveau de la saisie est conditionné par la définition du cycle de saisie. Dans ce cas une double analyse est effectuée. L'analyse standard d'abord et, si elle échoue, une relecture du caractère par le système de l'éditeur *ed_read_keycode* avec une nouvelle analyse.
Par le procédé de la touche alternative, il est possible d'échanger un symbole (par exemple: up) par une constante éditeur (ed_up) et ainsi éviter la relecture. Ce système est àconseiller lorsque l'on modifie le comportement "naturel" de l'éditeur.

D. *L'envoi de l'action*

Une fois l'action déterminée, elle est exécutée selon les instructions du prédicat *envoie.* Nous aurons encore àvoir comment les actions peuvent modifier le fonctionnement de la boucle de saisie. Avant de reprendre la position d'attente d'un événement, le système contrôle évidement si l'action ne correspond pas àune condition d'interruption. Pendant l'action, la clé d'origine est àdisposition par le prédicat *getlastkey*.

E. *Le contrôle de la boucle de saisie*

Comme évoqué plus haut la gestion des événements dépend du cycle de saisie. Lorsqu'un nouveau cycle de saisie est défini avec *nouvCycleSaisie* son nom est placé sur le stack *cycleSaisie* avec l'information sur l'utilisation de l'éditeur. En même temps les combinaisons clés-actions actives pour ce cycle sont chargés. Elles resteront actives jusqu'au chargement d'un autre cycle de saisie ou le déchargement du cycle de saisie courant. Ce déchargement s'effectue àl'aide de *finCycleSaisie.* Dans ce cas le cycle de saisie précédent est rechargé. Il est possible de définir un *postSaisie* lié au nom du cycle de saisie qui sera exécuté lors du déchargement du cycle et un *repriseCycle* exécuté lors du retour àun cycle donné.

F. *La gestion des clés actives*

Chaque cycle de saisie possède ses propres définition de relations clés-actions. Pratiquement, ces relations sont définies dans les fichiers "*.key". Ce sont des fichiers database de prolog chargé au moment de la définition du cycle de saisie (voir plus haut). La définition des clés se fait dans le code du programme *pekey*. Compilé et exécuté, c'est lui qui crée les fichier "*.key" Les arguments de la database *activeKey* sont:
- le nom du cycle de saisie,
- le nom de la touche (sgl(touche)) ou du groupe de touche (cf. référence) considéré,
- le nom de l'action déclenchée,
- nil ou la clé de substitution dont le système tiendra compte par la suite.

Par défaut, il faut définir un fichier par cycle de saisie défini. Il est possible d'attribuer un ensemble de commandes différent en liant, avant le chargement du cycle, un nom de fichier au nom du cycle par la commande *assert (Cycle, Fichier).* Pour défaire le lien, utiliser le prédicat *cancelKeyFile.*

A remarquer que les noms des fichiers doivent porter le nom du cycle pour des raisons historiques. Il serait possible de réduire *activeKey* àtrois arguments, considérants que les clefs actives sont celles qui sont chargées. Cela permettrait plus de souplesse encore!

G. *Les cycles de saisies principaux*

lexique: règle la saisie lors de l'affichage d'une fenêtre lexique. Il y a la possibilité d'accéder àdes cartes de connaissances àpartir du lexique.
view: visualisation dans un éditeur sans possibilité de modification.
wfk: attente d'une touche quelconque dans le prédicat waitforkey.
aidestat: semble n'activer que la touche return
lec: commande l'éditeur, dans le cas d'une édition en ligne (souvent utilisé dans les exercices, également dans boîte,...).
boxmenu: commande la fenêtre d'un menu.
hyperdeb: cycle de saisie spécifique au premier noeud d'une carte de connaissance. Il est important àcause du postSaisie défini sur cet identificateur de cycle.
hyper: commande la saisie dans les cartes de connaissances suivantes.

guidedeb: commande la première fenêtre du guide.
guide: commande les fenêtres guide suivantes.
pdwmenu: commande la saisie dans le menu déroulant.
button: commande la saisie entre les boutons.
interaction: est l'environnement de commande de l'interaction lors d'une question.
lacedit: commande le fonctionnement de l'éditeur propre àl'interaction lacune.
top: les commandes de base lors de l'entrée dans prof'expert.

Hypertexte et boucle de saisie centralisée

Afin de fonctionner correctement, il faut remonter àla boucle de saisie chaque foi qu'une saisie est nécessaire. Mais, le moteur hypertexte est, dans sa première forme une boucle dont on ne sort que lorsque l'on sort du premier noeud hypertexte.

Ainsi les deux "moteurs" sont incompatibles. Dans la mesure où la priorité est donnée à la saisie, le moteur hypertexte a été transformé en mode semi-automatique selon les principes suivant:

Lorsqu'une saisie est nécessaire (interaction, objectif, ...), le prédicat 'action' effectue toute les opérations jusqu'à la saisie. Cela implique que parmi les touches actives, il faut prévoir une touche de sortie, qui appelle le prédicat mot_hypReturn.

Lorsqu'il n'y a pas de saisie (énoncé, réponse) le prédicat 'action' fait lui-même appel, lui, au prédicat mot_hypReturn.

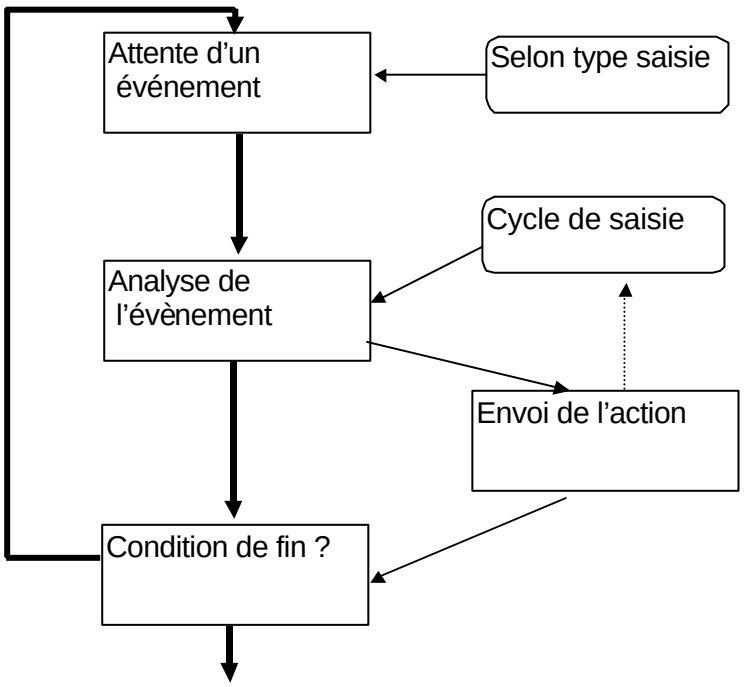


Fig 4: principe de la boucle de saisie

Pour le bon fonctionnement du cycle (figure 4), il faut définir au moins un événement débouchant sur l'action "fin du cycle de saisie".

Concernant l'exécution de l'action, la règle générale est qu'à la fin d'une action, il faut être remonté au niveau de la saisie.

Une action avec saisie procède d'abord à la modification du contexte de saisie puis effectue les actions avant la saisie (actions en attente).

Lorsque l'action déclenchée demande une saisie, la partie que le prédicat "envoie" déclenche directement n'est que la préparation à la saisie. La saisie n'est pas programmée autrement que par la modification du contexte de saisie. Ce nouveau cycle d'action aura bien entendu son événement échappatoire.

Les Fenêtres

La gestion des fenêtres pourra se faire à l'aide du prédicat interface *gestFenêtre(cmd_fenêtre, ListAttribut, ListAttribut) (I,I,O).* Le troisième argument est la validation du bon déroulement des opérations [att("valid","ok")]. Le premier argument est le type d'opération à effectuer, le deuxième sert à passer les paramètres de la fenêtre sur le modèle de la liste des attributs d'un noeud.

Les commandes disponibles sont:
- afficher une fenêtre (bien sûr),
- stocker les informations (complètes ou partielles d'une fenêtre),
- effacer une fenêtre
- basculer d'une fenêtre àl'autre.

Des éléments supplémentaires peuvent être demandés, tel le bouton de fermeture ou les barres de défilement, sur simple présence ou absence d'un attribut.

La database deffenêtre mémorise la définition des fenêtres utilisées. A une fenêtre peut aussi être lié un éditeur. Un attribut précise ce fait. Les éditeurs ouverts sont enregistrés dans edStat de la database - edit

Menu

Le menu possède sa propre boucle de saisie (enclenchée par menut). Les éléments de stack liés au menu sont distingués des autres événements à l'aide d'un flag supplémentaire.

database - pdwstate contenant le prédicat pdwstate
database - descrpdw contenant les prédicats descrpdw et mousestate

Objet interaction

Cet objet est constitué principalement d'un prédicat: interaction, qui marque les différents moments de l'exécution d'un exercice du point de vue de la configuration nécessaire aux différentes actions de l'utilisateur. Ce prédicat appelle une fonction particulière (flash, lacune, qrep, ...) pour chaque type d'exercice. Ces prédicats sont implantés dans des fichiers particuliers, un pour chaque type d'interaction. Ils contiennent en principe: un appel de début (qRep), un appel pour la préparation à la lecture (qRepRepo), un appel pour l'aide ponctuelle (qRepRevoir) et un appel pour la fin (qRepFin). Le type d'exercice 'flash', est un peu différent dans la mesure où la lecture des touches se fait directement au niveau de l'interaction. Dans ce cas le prédicat interaction a deux paramètres, interaction(type,touche).

La rétroaction ou feed-back (prédicat retroAction) est également organisé àpartir de ce module.

Objet experts

On fait appel aux experts par le prédicat callExpert(paramètre,ret). Le fonctionnement des règles dépend principalement de quatre types de faits donnés par les contextes experts(dom()), experts(did()), experts(psy()), experts(mode()).

Les différentes fonctions des experts, représentées par leur paramètre, sont:

établitEnoncé: Cette fonction est liée principalement àl'expert du domaine et au mode. Elle est appelée lors du chargement de la question. Son rôle est de:
- lire l'énoncé de la base (moteur hypertexte) ou le calculer,
- le transformer sous forme imprimable (l'interaction intervient aussi ici!).

fabriqueEnoncé: Certains modules existent pour fabriquer des énoncés automatiquement. Cette fonction les enclenchent lors de l'absence d'énoncé.

structure: Cette fonction, également liée àl'expert du domaine, peut être appelée par la fonction évalue de l'expert didactique. Elle met la réponse de l'utilisateur sous forme structurée. En mathématique, elle fait appel àun parser.

corrige: Est appelée par l'interaction lors de la saisie de la réponse pour contrôler la syntaxe de la réponse. Par exemple: format d'un nombre valide.

établitParamètre: Etablit toutes les informations pour la présentation de l'énoncé (temps d'affichage). C'est aussi cette fonction qui calcule la réponse et peut mettre en forme certaines méthodes de travail. Elle dépend de l'interaction, de l'expert didactique et de son mode.

évaluationInterm et évaluation: Analyse et évalue la réponse de l'utilisateur après saisie de la réponse.

actionDidactique: Cette fonction dépend de l'interaction et de l'expert didactique. Elle suit l'évaluation et modifie certains paramètres liés àla présentation des questions.

évaluationFinale: Evalue l'ensemble de l'exercice àla fin de l'exercice

Les travaux d'expertises importants sont réalisés par des fonctions particulières, contenues dans des fichiers spécifiques pour chaque type d'exercice ou pour chaque action générale (parser).

Objet générateur de messages

Le fonctionnement du générateur de message est assuré par un seul prédicat: constMSG(msglist,STRING). msglist est une suite de mots-clés qui permettent de construire un message. Dans l'état actuel, la plupart des messages sont déjàpréparés et l'action de constMSG revient à choisir un des messages au hasard avec parfois certaines indications supplémentaires: ton du message, nombre de fois ou ce type de message a été appelé (la quatrième fois où le système doit dire non, il dit nonnnn!). Il est possible d'associer àdes exercices ou des questions des messages particuliers. Ceux-ci sont dans des noeuds liés par un handle *msg*. Le noeud possède un attribut ton et un attribut sexe. Le lien possède un attribut particulier err_typ dont la valeur peut-être une réponse spécifique ou « err_typ » (message en cas de réponse fausse) ou « juste_typ » (message en cas de réponse juste). Il sera possible d'introduire d'autres valeurs qui permettront de graduer ces messages (réponse très fausse, plusieurs fautes successives, juste après plusieurs erreurs successives, etc.).

Annexe A: Cahier des charges (janvier 1992)

Description sommaire

Par rapport au système existant, la nouvelle version de Prof'Expert aura les particularité suivantes :

-intégration; il y aura moins de modules, les informations concernant les modes d'interaction seront intégrées dans les données,

-choix des exercices par objectifs pédagogiques,

-l'aspect apprentissage par "drill and practice" dans un contexte pédagogique explicite sera mis en évidence.

Les logiciels d'exercices, plus facilement réalisables et moins coûteux, peuvent assurer des apprentissages efficaces si certains concepts psychologiques sous-tendent leur utilisation.

Consignes générales

Chaque exercice, en plus d'un apprentissage technique, devrait :

-apporter une certaine ouverture en s'intégrant dans un cadre fonctionnel. Il faut assurer une certaine variété des sujets, veiller à la signification sociale, se méfier des stéréotypes.

-exercer, à travers l'interaction, des capacités cognitives de base (mémoire, mise en relation, induction, ...)

-contenir suffisamment d'informations pour assurer l'autonomie de l'apprenant.

Prof'Expert est un système malin et sympa (selon un journaliste de l'Impartial). Une réputation à maintenir !

Consigne pour la réalisation des expertises

Réalisation d'une carte des connaissances

Il s'agit d'une liste d'objectifs avec leur relation de dépendance. Dans une carte, il y a toujours un objectif principal (racine) et des sous-objectifs. Un exemple simple est le

mémento (racine) et les différentes règles (sous-objectifs). Chaque carte a un nom. Chaque objectif a un mémo (r1) et une description (Accord des adjectifs).

Certains objectifs sont intégrateurs de sous-objectifs, d'autres sont opérationnalisables (en particulier les feuilles de l'arbre).

Pour chaque objectif opérationnalisé on indiquera la façon dont on vérifie son acquisition (décision, test, exercice réalisé avec un certain taux de réussite).

Les experts doivent également préciser :

- la stratégie de progression envisagée (choix libre, progression, parcours imposé, ...)

-les contextes[1*] par défaut[2*]. Les principaux contextes sont le contexte d'aide (quelle est l'aide à disposition dans le cas général, comment cette aide se particularisera-t-elle ?) et le contexte d'interaction (y a-t-il une façon générale et commune de présenter les questions, analyser les réponses, générer les messages ?).

Modèle de l'élève

A la carte des connaissances, correspond la carte des savoirs de l'apprenant. C'est une partie du modèle de l'élève qui peut être complété par d'autres informations (style cognitif, noyaux de savoirs, etc. (cf. document IRDP, JFP,LOP à propos du calcul mental).

Réalisation des exercices

Un exercice est constitué d'une série de questions. Pour chaque exercice on donnera :

- son nom,
- l'objectif associé,
- un titre,
- une consigne,
- le cadre fonctionnel,
- une liste de paramètres (variables didactiques) et leur valeur par défaut (par exemple la difficulté d'une question),

---

[1] Contexte prend ici une signification technique. Le fonctionnement de PE est conditionné par des contextes (programmation par contexte). Un contexte est donné par une liste de "faits"*. Tous les termes marqués par un astérisque sont répertoriés dans un glossaire en annexe.

[2] PE est constitué d'"objets"*. Une question est un objet, un message est un objet. Chaque objet contient diverses informations : un texte, des indications sur la manière de s'afficher, des indications sur l'aide disponible, etc. Lorsque, pour un objet, une information fait défaut, PE utilise à sa place l'information correspondante de l'objet englobant.

- une stratégie de progression (en fonction des paramètres, au hasard, ...),
- un contexte d'aide,
- un contexte d'interaction (présentation, analyse des réponses, messages)

Chaque question sera caractérisée, au cas où les valeurs par défaut ne conviendraient pas, par :

- son nom,
- l'exercice dont elle fait partie,
- un générateur d'énoncé (en général un énoncé !),
- un générateur de réponse (en général une réponse),
- un titre,
- une consigne,
- le cadre fonctionnel,
- une liste de paramètres (variables didactiques) et leur valeur par défaut (par exemple la difficulté d'une question),
- une stratégie de progression (en fonction des paramètres, au hasard, ...),
- un contexte d'aide,
- un contexte d'interaction (présentation, analyse des réponses, messages)

Les énoncés sont constitué d'une information (stimulus) textuelle (hypertexte lié), et sont caractérisés par un cadre fonctionnel[3] et une valeur aux différents paramètres (difficulté de lecture, par exemple). Ils peuvent contenir des options d'interaction et d'aide.

Une réponse est une action ou un texte attendu. Elle peut être accompagnée d'alternatives qui seront prises en compte dans le contexte d'interaction.

Cadre fonctionnel et paramètres feront l'objet de descriptions particulières selon le sujet traité (voir document LOP sur les tests d'objectifs)

Système d'aide et messages

Aux actions des apprenants correspondent des réactions de la machine :

- messages d'erreur (voir àce propos les recommandations de Blavin),
- aide statique (indépendante du contexte, voir le programme approximation de l'addition réalisé pour l'Ecole normale de Neuchâtel),
- aide dynamique (dépendante de la question),
- aide interactive (avec boutons de marquage, etc.).

[3] Un cadre fonctionnel est le contexte (au sens courant du terme) est présenté une notion (aspect historique, ludique, ...). Les thèmes pourraient être repris des critères de classement des mots de Orthobase. Voir le document: Vers une définition de cadres fonctionnels.

L'aide introduit un aspect tutoriel. L'hypothèse sous-jacente est que, en réaction à la pression exercée par le rythme des exercices, la consultation libre d'informations donne lieu àdes apprentissages durables.

Evaluation

Aspect cumul des connaissances : une carte des savoirs, pendant de la carte des connaissances, sera tenue àjour pour chaque apprenant.

Aspect qualitatif et métacognitif : pour chaque exercice des indications plus qualitatives sur les réponses données et les cheminements suivis pourront être enregistrés (possibilité de marquage sous forme de boutons dans le système d'hyperaide). La forme de présentation de cette information reste àtrouver.

Il faut utiliser l'ordinateur pour décrire nos processus et nous permettre de les analyser et les comparer.

Expertises

En résumé les expertises àréaliser sont :

- dans la discipline : recherche de la réponse juste et des erreurs typiques,

- au niveau psychopédagogique : type d'interaction en fonction de la question et du modèle de l'élève, ajustement du modèle de l'élève en fonction des réponses données.

Planning

En ce qui concerne les expertises les grandes étapes sont les suivantes :

Eté 92 : Classification des exercices existants selon le schéma adopté. Examen du problème de l'évaluation. Choix de structures définitives qui tiennent compte de nouveaux types d'exercices.

Octobre 92 : Les différentes structures sont définitivement arrêtées.

Noël 92 :Réalisation de jeux d'essais pour tous les types d'exercices. Mise au point de l'existant dans la nouvelle forme. Etablissement des listes de catégorie (messages, styles cognitifs, types d'interaction, ...).

A partir de janvier 93, les modules sont testé et enrichis, le manuel mis en chantier.

Les Experts ont tout loisir d'inventer des types d'exercices et d'imaginer des types d'interaction. Dans la mesure du possible des maquettes seront chaque fois réalisées. Il faudra parfois trancher ... (cf. les trois critères en bas de page 1)

Dossier de programmation

Généralité

Prof'Expert est un moteur d'inférence avec un cycle principal dont la fonction essentielle est de modifier le contexte. Les cycles secondaires sont conditionnés par le contexte.

Les données utilisées par PE (faits) sont de 5 types :

- Les structures de base (carte des connaissances) particularisent PE àune application particulière. Elles peuvent être compilées.
- Les exercices (Database).
- Une base d'informations de base (cf. PROFTEXT).
- Les messages (cf. EDITMSG, GENERE).
- Les aides structurées (cf. EDITAIDE).

Le fonctionnement de Prof'Expert est entièrement conditionné par les données. Le programme est au service des données. Les données sont au service des apprenants.

Fonctionnement du programme

Opération àl'enclenchement

setup 1
- établissement des chemins d'accès pour le suivi et les données,
- choix des couleurs.

setup 2
- chargement d'une carte des connaissances (si non compilée),
- établissement du contexte par défaut (interface, aide, ...).

Choix d'un exercice

Si le système est en mode suivi : chargement de la carte du savoir

- choix d'un objectif (automatique ou selon demande)
- choix d'un exercice (automatique ou selon demande)

Fonctionnement du moteur

- adaptation du contexte (point de backtrack*)
- choix d'une question (selon divers modes)
- adaptation du contexte (PPC)
- génération d'un énoncé (expertise possible)
- génération de la réponse et paramètres divers (erreurs typiques ...) (expertise)
- interaction (utilisation du contexte d'aide)
- message (expertise)

- mise àjour cartes des savoirs (selon indice de progression) (expertise)
- enregistrement de la réponse
- rétablissement du contexte par défaut

Fin d'un exercice ou d'une session

- Evaluation de la session
- Sauvetage de l'exercice en cours
- Mise àjour du suivi

Les versions de Prof'Expert

Pour satisfaire le parc des machines existant, le système devra fonctionner :

- avec fichiers d'exercices séparés ou avec des databases,

- en mode texte, texte avec souris, graphique,

- sur des machines isolées, en réseau, PEM.

Prof'Expert doit être un système maniable, utilisable sur un parc de machines assez étendu (PC des élèves !).

Les travaux associés

Diverses réalisations permettront d'assurer le suivi de Prof'Expert tant du point de vue technique que pédagogique.

- Aide et information par télématique.

- Possibilité de télécharger des exercices.

- Etude d'autres interfaces (Windows)

Planning

Les grandes étapes sont :

1. jusqu'en été 92: Préparation des structures et des bases de données. Test et complément aux bibliothèques. Réalisation de ESTIME et CALCUL selon le principe unifié. Installation de la souris. Structure de base en mode graphique (adaptation de formule).

2. octobre 92 Transformation des données existantes. Esquisse du système de prise de données et du gestionnaire du suivi.

3. Noël 92 Mise au point des bases de données et du mécanisme général. Diffusion d'une première version (réseau et PEM).

4. Eté 93 Adaptation et mise au point du système. Mise au point du système pour la prise de données. Réalisation du générateur de message et du gestionnaire du suivi en collaboration avec des élèves du CPLN (travail de diplôme). Réalisation de la partie technique du manuel.

Les droits du concept Prof'Expert appartiennent àl'association (en voie de constitution) "apprentissages de base et ordinateur àtout âge" (ABORDAGE).

Un programme parfait est un programme obsolète (nième loi de Murphy)

Glossaire

fait : une donnée enregistrée par le système en cours de travail. Par exemple : *élève rapide*. L'ensemble des faits à un moment donné constitue le contexte (mémoire de travail).

règle : une opération qui modifie un contexte donné.

contexte : ensemble des faits àun moment donné.

moteur d'inférence : cycle de modifications du contexte en fonction du contexte et des règles et des actions de l'utilisateur.

backtrack (retour en arrière) : technique de contrôle de systèmes informatiques qui en cas de problème, rétabli les données de départ et procède àune nouvelle tentative.

objets : éléments caractérisés par plusieurs attributs.

attribut : caractéristique particulière d'un objet qui peut prendre plusieurs valeurs (dans une liste ou par calcul). Une valeur par défaut est attribuée àchaque attribut.

hypertexte : base de données (textuelles) organisées en réseau. Chaque sommet du réseau est un noeud. A chaque noeud sont associées différentes informations ou procédures.

hyperchamp : partie d'un texte qui permet d'activer le passage à un autre noeud de l'hypertexte (appelé aussi un bouton).

noeuds : sommet d'un réseau.

L.-O. Pochon/ janvier 92

Annexe B: Hypertexte

Prof'Expert s'organise selon une structure d'hypertexte. Les hypertextes se trouvent aux confluents de divers travaux, idées, recherches et nécessités liés à l'utilisation de systèmes automatiques de traitement de l'information: systèmes d'aides, documentation complexe, édition structurée, création littéraire.

Cette architecture a été choisie afin de pouvoir profiter d'un mode de navigation qui semblait approprié aux objectifs du projet. Il possède de nombreuses possibilités d'intégration de structures diverses, il bénéficie d'acquis théoriques sur la lecture par ordinateur (des travaux qui permettent de calibrer l'information et son accessibilité). Il permet également d'intégrer au système des règles de sélection qui permettent de personnaliser la navigation en fonction du type d'utilisateur.

Notions de bases

La notion la plus simple est celle d'unité d'information (UI). C'est un texte (ou un schéma) àvaleur informative qui constitue un segment d'une information plus complète, qui compréhensible moyennant une charge cognitive faible. Une unité d'information pourra être une explication d'un procédé, une règle de grammaire, une loi, etc. Une consigne ou l'énoncé d'une exercice, un message constituent également des UI. Une unité d'information n'est pas liée à un dispositif informatique, bien que certaines contraintes (page écran) existent qui peuvent en modeler la structure.

La deuxième notion importante est la notion de concept. Un concept peut être perçu de diverses manières. Il peut consister en une unité linguistique (identification des mots et des concepts), cela peut être une construction mentale abstraite (réseau sémantique) qui possède valeur de consensus dans un milieu donné (concept de réussite à un examen) ou une unité structurale pragmatique. Par exemple, et ce sera l'utilisation la plus fréquente dans Prof'Expert le découpage d'une matière d'enseignement en différents objectifs ou contenus.

L'ensemble des concepts organisent un certain domaine de connaissance.

Concept, unités d'information prennent parfois leur sens courant, parfois un sens ad hoc pour uniformiser le système.

Concept et unité d'information sont imbriqués dans la mesure où chaque unité d'information peut être résumée par les concepts qui y sont inscrits ou décrits. Par exemple, l'unité d'information suivante:

"Pour accorder le participe passé, il faut tenir compte du sujet de la phrase, du complément d'objet direct et de l'auxiliaire utilisé"

contient les concepts suivants: participe passé, sujet du verbe, phrase, complément d'objet direct, auxiliaire. Ces concepts constituent le contenu conceptuel de l'UI.

Le contenu conceptuel d'une UI peut également apparaître sous la forme de boutons ou de concepts cachés (action des touches).

Par ailleurs un certains nombre de concepts font appels àcette unité d'information (par exemple: accord des participes passés). Ces concepts sont les descripteurs de l'UI.

Pour un concept donné, le nombre d'UI dont il est le descripteur s'appelle la valence du concept. Ce nombre est également le rendement du concept (Re(c)). La valence d'une UI est le nombre de ses descripteurs.

Très souvent une UI n'a qu'un seul descripteur.

Soit Uj une unité d'information. D(Uj) = {cj1, cj2, ..., cjk} ses descripteurs. Soit ci un concept et U(ci) = {ui1, ... uin} l'ensemble des unités d'informations dont il est le descripteur:

Re(Ci) = rendement de ci = valence de ci = n

N = nombre de concepts pondéré = Ó Re(Ci)

Di(Uj) = disponibilité d'une UI = #D(Uj) / N

D = disponibilité moyenne = Ó Di(Uj)/nombre d'UI

corr(c1,c2) = corrélation des concepts c1 et c2 = #U(c1) ^ U(c2) /(re(C1)+re(c2))

Ces indices, avec le nombre de boucles et la profondeur des annotations (voir plus loin), donnent une indication sur la lisibilité de l'hypertexte. Ces indices peuvent être particularisés àdes contextes particuliers au cas où une UI apparaîtrait avec différentes variantes.

Notion de relation

A l'image des réseaux sémantiques, les concepts et les UI sont reliés par des relations. Ces relations pourront avoir divers noms. La plus importante étant celle de nom "descripteur" qui lie un concept à une UI. On distingue les relations passives et les relations actives. Parmi les premières, il y a les relations fixes et les relations calculées.

Relations passives

fixes: C: calcul_mental -> UI: exercice_1
C: calcul_mental -> UI: ui_procédés_de_calcul_mental

Le première relation est de nom exercice, la deuxième de nom descripteur.

calculées: Ce sont les relations qui existent mais qui sont particularisées en fonction de l'environnement. C'est le cas dans le guide. C'est aussi le cas lorsqu'une unité d'information est constituée de toute pièce (génération d'énoncé ou fabrication de réponse). Dans cette catégorie on peut faire mentionner la notion de parcours filtrés (webs) où le parcours des unités d'informations se fait avec un point de vue particulier (restriction aux énoncés faciles ou aux informations rédigées de façon théorique).

Les relations actives sont celles qui sont générées par le système selon divers procédés. Dans le cas de prof'Expert, l'accès au lexique peut constituer un exemple de relation active. Dans ce cas le système considère la racine du mot et cherche une UI dont le nom est approchant. D'autres techniques sont proposées par exemple, le procédé le la distance: recherche d'un mot correspondant au concept d'originalité (farfelu a sur cet axe une valeur de 3,5 sur 10). La recherche d'une information peut aussi découler d'une expertise particulière àl'aide de règles de production, par exemple ou par une procédure classique.

type d'UI: objectif, exercice, question, donnée, image, relais

handle: <concept>, aide_stat, méthode, hlp_guide, question, énoncé, réponse, doncpl, slideshow, ctrlguide, guide, msg, distracteur

liens passifs:
fixes: question, énoncé, réponse, descripteur, exercice, guide
calculés: question, exercice, descripteur, énoncé, réponse, guide

liens actifs: lexique

Annexe C: Description des types d'exercices

Cette description détaille un élément charnière entre l'analyse et la programmation. Un type d'exercice est constipé de briques qui sont agencées de façon diverse selon le domaine considéré.

Pour chaque type d'exercice on utilisera les rubriques suivantes:

Format des données: Il s'agit du format utilisé.

Contextes: On signalera ici les contextes spécifiques ou obligatoires au type d'exercice, sachant qu'un certain nombre de contextes sont très généraux et peuvent ou non figurer selon les souhaits des créateurs de l'exercice: interaction(reprise()), contrôle(remise()), contrôle(ordre(_)), interaction(feed_back ()),suivi().

Attribut particulier d'un exercice ou d'une question: l'attribut niveau est général de même que objectif.

Consigne: Sauf indication contraire elle constitue le content du node exercice.

Recherche de l'énoncé: directe ou calculé.

Mise en forme de l'énoncé: callExpert(établitEnoncé,_) effectue ce travail particularisé selon l'exercice.

Recherche de la réponse: Directe ou calculée de diverses manières, en principe lors de l'établissement des paramètres.

Etablissement des paramètres: Par callExpert(établitParamètres,_).

Mise en forme de la réponse: Destinée au coup d'pouce, cette réponse sous forme imprimable est établie en même temps que la réponse.

Affichage de l'énoncé: Type de fenêtre utilisée (dimension, position) et mode d'affichage.

Prise de la réponse: Manière dont l'utilisateur entre la réponse.

Analyse et mise en forme de la réponse (syntaxique) : La réponse peut être prise telle quelle ou structurée (callExpert(structure,_)).

Evaluation de la réponse: Deux appels aux experts peuvent intervenir, callExpert(évalInterm,_) (analyse de la réponse) et callExpert(évaluation,_) (jugement).

Format des résultats: Les informations enregistrées de façon systématique sont le nom de la question, le temps de travail, le concept, le nombre d'appel au coup d'pouce, nombre d'appel àla correction (nombre d'essais).

Action didactique: Par callExpert(actionDidactique,_).

Evaluation finale: A la fin de l'exercice elle est appelée par callExpert(évalFinale). Un calcul de l'indice de réussite global (moyenne statistique des indices par concept réalisé) et un résumé est élaboré, contenant notamment le nom de l'exercice, le statut en fin d'exercice (exercice repris ou nouveau, arrêté ou fini), le temps mis pour faire l'exercice.

Aide statique: En principe tous les exercices ont un certain nombre de liens fixes: aide_stat, méthode, hlp_guide. Ces liens peuvent aussi est particularisés question par question.

Aide dynamique: aide particulière, calculée en fonction des données.

Actions diverses: En particulier le coup d'Pouce.

Mathématique

ESTIME

Format des données: exp (expression arithmétique).

Remarque: Dans les versions 2 cette expression contient la réponse et l'indice de difficulté. Ces deux valeurs sont calculées dans la version 3

Contextes: interaction(type(flash)), position(centré, aléatoire), experts(dom(calcul)), experts(did(estime)), tempo(Nombre). Le contexte contrôle(remise(perpète)) associé à un exercice constitué d'une seul question dont l'énoncé est nil, provoque une présentation sans fin de questions fabriquées par le système. Dans ce cas, deux contextes de contrôle sont nécessaire: paramètre(Nombre de termes, Grandeur des termes) et s_paramètre(Domaine de nombres, Opération). A voir pour un contexte d'attente entre deux questions.

Consigne: dans le content de exercice (2 lignes). Paraît avant les flash. Si la consigne de la question n'est pas nil, cette consigne est également flashée avant l'énoncé (utile lorsque l'exercice n'est pas de type flash mais contient des questions de ce type.

Recherche de l'énoncé: Directe ou calculé.

Mise en forme de l'énoncé: l'expert du domaine 'calcul' transforme la notation polonaise en expression AOS.

Recherche de la réponse: calculée par l'expert didactique estime.

Etablissement des paramètres: callExpert_did(établitParamètres,_) calcule l'indice de difficulté, calcul du temps et établissement de la réponse. TEMPS = TEMPS DE BASE * NT * log (2+DIFF) avec NT nombre de termes et DIFF difficulté.

Mise en forme de la réponse: expert du domaine: calcul, voir ci-dessus.

Affichage de l'énoncé: dans une fenêtre centrée ou aléatoire selon le contexte. Le temps de présentation est limité.

Prise de la réponse: dans un prompt centré de format calculé (l'énoncé a disparu). L'analyse de la syntaxe est effectuée au cours de la frappe (permis -; 0-9, , ou .) par callExpert(corrige,_).

Analyse et mise en forme de la réponse (syntaxique) : faite par l'expert du domaine calcul (et son parser associé). Cet expert émet ses propres messages. En principe dans cette interaction, il ne devrait pas y avoir de fautes de syntaxes.

Evaluation de la réponse: Calcul de deux indices: précision P et rapidité R (ces deux valeurs sont comprises entre 0 et 1), enregistrement du concept.

Format des résultats: Les information spécifiques sont: précision(P), rapidité(R), indice global = 100 (1- (P+R - P*R))

Action didactique: modification du temps de base.

Aide dynamique: sans

Actions diverses: "revoir", diminution du tempo.

LACUNE/enigme

Format des données: expr (string).

Contextes: interaction(type(lacune)), mode(enigme), experts(dom(lacune)), experts(did(lacune)), tempo(Nombre) (temps de revoir)

Attributs particuliers: Dans la question: remp : caractère de remplacement. Dans l'énoncé: window et lisibilité (il pourrait y avoir plusieurs énoncés de lisibilité (différentes).

Consigne: Dans le node question.

Recherche de l'énoncé: directe àpartir de la question.

Mise en forme de l'énoncé: l'expert du domaine 'lacune' et son mode remplace les caractères souhaités et calcul le nombre de remplacements.

Note: certains caractères ne sont pas remplacés (\32, \196, \179, =, ...).

Recherche de la réponse: directe, c'est l'énoncé.

Etablissement des paramètres: calcul du nombre de termes remplacés.

Mise en forme de la réponse: string contenu dans énoncé sans certains caractères de commande (crochets).

Affichage de l'énoncé: Dans une fenêtre selon window contenu dans les attributs avec printname ou name

Prise de la réponse: dans un prompt au bas de l'écran. Tous les caractères sont permis. Après chaque proposition de l'utilisateur, l'énoncé est remis àjour (opération de type 'mise àjour').

Analyse et mise en forme de la réponse: sans.

Evaluation de la réponse: après chaque saisie. Le nombre d'essais sert de base pour l'indice de performance.

Action didactique: diminution du tempo.

Evaluation finale: Calcul de l'indice global à partir de nombre de remplacement et le nombre d'essais pour compléter les lacunes.

Aide dynamique: sans.

Actions diverses: revoir selon tempo

LACUNE/crochet

C'est un sous mode du mode enigme. Seuls les expressions entourées d'un crochet sont remplacées par le texte de remplacement.

Note pour la réalisation des textes: ils doivent être tout d'abord rédiger sans les crochets. Puis les crochets sont ajoutés. Le texte de remplacement doit être plus court que les textes àremplacer.

Analyse et mise en forme de la réponse: Une analyse minimale est réalisée. Les espaces ne sont pas comptés et l'équivalence numérique est admise. Des paramètres supplémentaires pourraient diriger plus précisément cette correction.

LACUNE/choix

Les particularités par rapport au cas précédent sont les suivantes.

Contextes: interaction(type(lacune)), mode(choix)

Attributs particuliers: Dans la question: aremp: mots à remplacer (chaîne avec séparateur |) remp : texte de remplacement. Dans l'énoncé: window et lisibilité (il pourrait y avoir plusieurs énoncés de lisibilité différente).

Etablissement des paramètres: calcul du nombre d'expressions remplacées (marquées entre crochets).

Prise de la réponse: dans un éditeur affichant l'énoncé.

Evaluation de la réponse: lors de l'action d'une touche particulière (F10). Le texte est mis à jour par l'utilisateur lui-même. Le nombre de fois où cette touche est actionnée sert de base àl'évaluation.

PROBLEME/QCM/MULTIR

Ce type d'exercice peut reprendre les données du français (ACCORDE) où alors faire appel plus explicitement àdes expertises mathématiques.

Format des données (énoncé et réponse): expr (string); expr2(string, liste de remplacement, défaut); exp(EXP), probleme(string,liste de variables à remplacer, liste de valeurs), liste de telles expressions. Des données complémentaires (cas des exercices avec calcul) sont contenu dans un node lié au node exercice par un lien de nom donCmpl.

Contextes: interaction(type(qrep)), experts(dom(texte| calcul), experts(did(texte| calcul| multir), experts(mode(proportionalité| calcul), interaction(tolérance (prct(N)| sans| frac_seule| f_irréd| f_autorisée), interaction(mode(normal| qcm(N))

Note: les qcm sont établis à partir des données liées par le lien de nom distracteur attaché à la question ou à l'exercice. L'expert didactique multir présuppose des réponses sous forme de qcm, plusieurs réponses sont attendues. Le contrôle est donc en mode reprise et l'énoncé est remis àjour.

Attributs particuliers pour les énoncés: window, lisibilité (il peut y avoir plusieurs énoncés de lisibilité différentes).

Attributs particuliers d'un exercice ou d'une question: window.

Consigne: dans le content de exercice (2 lignes). Complément dans le content de question.

Recherche de l'énoncé: directe àpartir de la question.

Mise en forme de l'énoncé: utilisation directe des expr, avec remplacement dans le cas des expr2, et certains calculs intermédiaires dans le cas problème.

Recherche de la réponse: directe, plusieurs réponses peuvent être possibles. Une réponse peut être une formule générale qui est évaluée en utilisant les données du problème et des données complémentaires.

Etablissement des paramètres: Selon le mode de l'expert, une méthode de résolution peut-être établie.

Mise en forme de la réponse: utilisation directe des expr, dans le cas des listes les éléments sont séparés par des point-virgules.

Affichage de l'énoncé: Dans une fenêtre de dimension 5x80 ou selon l'attribut de l'exercice ou de l'énoncé.

Prise de la réponse: Dans un prompt en mode normal. A l'aide d'un menu dans le mode qcm. En général, une réponse simple est attendue. En cas de réponses constituées de plusieurs items, ceux-ci sont séparés par des ';'. L'opération de 'mise à jour' peut exister si le contexte le demande avec dans ce cas remise àjour de l'énoncé.

Analyse et mise en forme de la réponse (analyse syntaxique): Structuration de la réponse en EXP.

Mise en forme du feed_back: concaténation de l'énoncé et de la réponse dans le cas des expr, remplacement dans les cas des expr2. Dans les cas où la réponse est de type expr2, on procède àune mise en forme de cette expression.

Evaluation de la réponse: par juste ou faux. Si l'option reprise ou remise est enclenchée, chaque réponse données par l'utilisateur est cataloguée.

Format des résultats: Le nombre de réponses justes ou fausses est pris en compte.

Action didactique: selon le experts(mode())

Evaluation finale: pourcentage de juste par rapport au nombre de tentative ventilé par concept est la spécificité de cette évaluation. On notera qu'il y a la possibilité en fin d'exercice de reprendre les questions où il y a eu erreur.

Aide dynamique: parcours guidé ou information particularisées aux données de la question.

ELEUSIS

Cette interaction, qui demande de trouver une suite de résultat, reprend le type de l'interaction précédente avec interaction(mode(eleusis)), experts(did(eleusis)) et experts(mode(suite0| suite1|suite2|suite3|suite4) chacun de ces derniers paramètres représentant un mode calcul. Le format des données est en principe formule(Expression de transformation, Liste des valeurs initiales). La liste des valeurs peut contenir une transformation auxiliaire. L'option interaction(reprise(avec)) est recommandée pour donner un sens àce type d'exercice.

FORMULE

Cette interaction, qui demande de trouver une formule donnant un résultat donné, reprend également l'interaction QRep avec interaction(mode(formule), experts(did(formule)). L'option interaction(reprise(avec)) est aussi recommandée pour donner un sens à ce type d'exercice. Le format des données est en principe formule(Expression de transformation, Liste des valeurs initiales). Cette liste contient souvent les valeurs sous la forme d'un intervalle des valeurs parmi lesquelles une valeur sera tirée au hasard. Les questions ont trois attributs particuliers: autorisé, obligatoire, interdit. Les valeurs de ces attributs sont des chaînes listées de symboles.
L'utilisation de ces valeurs est la suivante lors de la correction de la réponse de l'élève. Le système teste d'abord que tous les symboles obligatoires sont présents. Puis, il regarde si les valeurs appartiennent à des domaines autorisés (N, Z, D, Q, variable, répétition) et ne sont pas interdits en tant qu'élément. Tout les autres signes non interdits sont autorisés!

CALCUL

Format des données: exp (expression arithmétique).

Contextes: interaction(type(flash)), position(centré, aléatoire), experts(dom(calcul), experts(did(calcul), interaction(tempo(50)), tolérance (voir PROBLEME). Le contexte contrôle(remise(perpète)) associé à un exercice constitué d'une seul question dont l'énoncé est nil, provoque une présentation sans fin de questions fabriquées par le système. Dans ce cas, deux contextes de contrôle sont nécessaire: paramètre(Nombre de termes, Grandeur des termes) et s_paramètre(Domaine de nombres, Opération). A voir pour un contexte d'attente entre deux questions.

Consigne: dans le content de exercice (2 lignes). Paraît avant les flash. Si la consigne de la question n'est pas nil, cette consigne est également flashée avant l'énoncé (utile lorsque l'exercice n'est pas de type flash mais contient des questions de ce type.

Recherche de l'énoncé: directe ou calculé.

Mise en forme de l'énoncé: l'expert 'calcul' transforme la notation polonaise en expression AOS.

Recherche de la réponse: calculée par experts(did(calcul)).

Etablissement des paramètres: On calcule l'indice de difficulté, le temps en même temps que l'établissement de la réponse. Le système recherche les différentes voies menant à la solution et qui seront utilisées comme guide. Pour certains calculs la formule: TEMPS = TEMPS DE BASE * NT * DIFF / 200 avec NT nombre de termes et DIFF difficulté.

Affichage de l'énoncé: dans une fenêtre centrée ou aléatoire selon le contexte. Temps de présentation limité.

Prise de la réponse: dans un prompt centré de format calculé (l'énoncé a disparu). L'analyse de la syntaxe est effectuée au cours de la frappe par callExpert(corrige,_)

Analyse et mise en forme de la réponse (syntaxique) : Avec le parser liée au domaine calcul.

Evaluation de la réponse: réponse juste ou fausse (àla tolérance près).

Format des résultats: On garde l'information juste ou faux.

Action didactique: modification du temps de base, message à l'utilisateur en cas d'utilisation répétée du coup d'pouce, modification des énoncés dans le cas "perpète"!.

Aide dynamique: guide présentant différentes façons de décomposer les calculs.

Actions ponctuelles: diminution du tempo.

FICELLES

Cette interaction reprend les principaux paramètres de calcul en prenant comme interaction(mode(multie)) et comme expert(mode(ficelle1|ficelle2)). Elle propose une chaîne de calculs, un nouveau résultat s'obtenant à partir du résultat précédent (des deux résultats précédents pour ficelle2) et du calcul proposé.

Résumé des types d'exercice

Type	Interaction		Experts		
	Type	Mode	Expert dom	Expert did	Expert mode

CALCUL	flash	-	calculm[4]	calcul	[calcul]
ESTIME	flash	-	calculm	estime	
FICELLE	flash	multie	calculm	calcul	ficelle1-2
FORMULE	qrep	formule	calcul	formule	
ELEUSIS	qrep	eleusis	calcul	eleusis	suite0-4
QCM	qrep	qcm(N)	texte	texte	
MULTIR	qrep	qcm(N)	texte	multir	
PROBLEME	qrep	normal	calcul	calcul	[proportionnalité, système2x2]
ENIGME	lacune	enigme	lacune	lacune	
LACUNE	lacune	choix	lacune	lacune	
LACUNE	lacune	crochet[1]	lacune	lacune	

Français

ACCORDE/QCM/MULTIR

Format des données (*énoncé ou réponse)*: expr (string); expr2(string, liste de remplacement, défaut); probleme(string, liste de remplacement, liste de valeurs), liste de telles expressions..

Contextes: interaction(type(qrep)), experts(dom(français)),experts(did(français| multir)), interaction(tolérance (majus| minus| sans| ortho| termin), interaction(mode(normal| qcm(N)), experts(mode(ppassé).

Note: les qcm sont établis à partir des données liées par le lien de nom distracteur attaché à la question ou à l'exercice. L'expert didactique multir présuppose des réponses sous forme de qcm, plusieurs réponses sont attendues. Le contrôle est donc en mode reprise et l'énoncé est remis àjour.

Attributs particuliers: pour les énoncés window, lisibilité (il peut y avoir plusieurs énoncés de lisibilité différentes).

[4] Conjonction de l'interaction: flash et de l'expert du domaine primitif: calcul.

Attributs particuliers d'un exercice ou d'une question: window.

Consigne: dans le content de exercice (2 lignes). Complément dans le content de question.

Recherche de l'énoncé: directe àpartir de la question.

Mise en forme de l'énoncé: utilisation directe des expr, avec remplacement dans le cas des expr2, avec experts(dom()) dans le cas probleme.

Recherche de la réponse: directe, plusieurs réponses peuvent être possibles.

Etablissement des paramètres: selon le mode de l'expert, une information d'aide (participe passé) peut être établie.

Mise en forme de la réponse: utilisation directe des expr, dans le cas des listes les éléments sont séparés par des virgules.

Affichage de l'énoncé: dans une fenêtre 5x80 sauf si window figure dans les attributs de l'énoncé, la question ou l'exercice.

Prise de la réponse: dans un prompt dans le cas normal. A l'aide d'un menu pour le mode qcm. En général, une réponse simple est attendue. En cas de réponse multiple, séparation par des ','. L'opération de type 'mise àjour' est effectuée, si le contexte le demande avec dans ce cas remise àjour de l'énoncé.

Analyse et mise en forme de la réponse (analyse syntaxique): En général la réponse est simplement décomposée en 'liste'.

Mise en forme du feed_back: concaténation de l'énoncé et de la réponse dans le cas des expr, remplacement dans les cas des expr2. Dans les cas où la réponse est de type expr2, on procède àune mise en forme de cette expression.

Evaluation de la réponse: sans reprise: juste ou faux (avec message différencié selon expert). Dans le cas avec reprise, le nombre d'essais fera fois.

Format des résultats: 'revoir', réponse juste ou fausse

Action didactique: selon experts(mode()). Dans le cas de ppassé, affichage du pp en cas d'erreur de racine.

Evaluation finale: pourcentage de juste par rapport au nombre de tentative. On notera la possibilité en fin d'exercice de reprendre les questions où il y a eu erreur.

Aide dynamique: parcours guidés et informations contextualisées avec les données de la question.

ACCORDE/TABLEAU

Cette interaction, qui demande de trouver un tableau de résultat, reprend le type de l'interaction précédente avec interaction(mode(tableau(données du tableau))). L'option interaction(reprise(avec)) est recommandée pour donner un sens àce type d'exercice. Dans ce cas le tableau sera remis àjour.

LACUNE/enigme

Ce type d'exercice est quasiment identique à celui de mathématique. Seuls les caractères àne pas remplacer diffèrent: - ' . , ! ?

LACUNE/crochet

C'est un sous mode du mode enigme. Seules les expressions entourées d'un crochet sont remplacées par le texte de remplacement. Contrairement au cas mathématique, l'alignement n'est pas conservé. Attention, le mode crochet de la version 2 devient choix!

LACUNE/mot

Les particularités par rapport au cas précédent sont les suivantes.

Contextes: interaction(type(lacune)), mode(mot)

Attributs particuliers: Dans la question: aremp: mots à remplacer (chaîne avec séparateur |) remp : texte de remplacement. Dans l'énoncé: window et lisibilité (il pourrait y avoir plusieurs énoncés de lisibilité différente).

Etablissement des paramètres: calcul du nombre de mots remplacés

Prise de la réponse: dans un éditeur affichant l'énoncé.

Evaluation de la réponse: lors de l'action d'une touche particulière (F10). Le texte est mis àjour par l'utilisateur lui-même. Le nombre de fois où cette touche est actionnée sert de base àl'évaluation.

LACUNE/syllabe

C'est comparable au mode mot. Simplement les textes àremplacer sont arbitraires.

LACUNE/choix

Idem sauf que ce sont les textes entre crochets qui sont remplacés par un texte de remplacement. Ce mode était l'ancien mode crochet.

FLASH_VOC

Format des données: Enoncé: expr (string); expr2(string, amorce, "");
Réponse: idem ou par défaut égale àl'énoncé.

Contextes: interaction(type(flash)), experts(dom(français)), experts(did(flash_voc)), position(centré, aléatoire), tempo(valeur), paramètre(min/max).

Attributs particuliers dans les données (énoncé): lisibilité (LIS: nombre qui sert avec le tempo àdéterminer le temps du flash) et fréquence (non utilisé)

Consigne: dans le content de exercice (2 lignes). Paraît avant les flash. Si la consigne de la question n'est pas nil, cette consigne est également flashée avant l'énoncé (utile lorsque l'exercice n'est pas de type flash mais contient des questions de ce type.

Recherche de l'énoncé: directe àpartir de la question.

Mise en forme de l'énoncé: utilisation directe des expr, avec remplacement dans le cas des expr2

Recherche de la réponse: directe, si pas de réponse, l'énoncé sert de réponse

Etablissement des paramètres: calcul du temps du flash: TEMPS = TEMPS de BASE * sqrt(LIS/8)

Mise en forme de la réponse: utilisation directe des expr.

Affichage de l'énoncé: dans une fenêtre flashée centrée ou de position aléatoire.

Prise de la réponse: dans une fenêtre au centre, les lettre tapées fausses ne sont pas affichées. Au cas où une amorce existe, elle est déjàaffichée.

Analyse et mise en forme de la réponse (syntaxique) :sans

Evaluation de la réponse: juste (aucune lettre tapée fausse) ou faux

Format des résultats: réponse juste ou fausse

Action didactique: ralentissement, accélération, etc.,

Résumé des types d'exercices

Type	Interaction		Experts		
Interaction	Type	Mode	Expert dom	Expert did	Expert mode
FLASH-VOC	flash		français	flash_voc	
TABLEAU	qrep	normal	français	français	
QCM	qrep	qcm(N)	français	français	[ppassé]
MULTIR	qrep	qcm(N)	français	multir	
ACCORDE	qrep	normal	français	français	[ppassé]
ENIGME	lacune	enigme	lacune	lacune	
LACUNE	lacune	choix| syllabe| mot	lacune	lacune	

Annexe D: Organisation des données

La liste donnée ci-dessous décrits la structure des différents noeuds de l'hypertexte. Elles ont reçus le nom d'objet dans la mesure où certaines des informations constituent des éléments actifs dans la commande de procédures ou de règles.

Objet: entrée (unique)
attribut: name = ui_entrée
attribut: type = objectif
content: text(REF)
linklist:

nom	handle	pointe sur
descripteur	ui_entrée	première UI

Objet: élément de connaissance
attribut: name = ui_<concept> * (obligatoire)
attribut: printname = <string>
attribut: type = objectif *
attribut: variable = <nom> (répété)
attribut: contexte =
content: text(REF)
linklist:

nom	handle	pointe sur
descripteur	<concept>	élément d'information
		relais de slideshow
exercice	<concept>	exercice
slideshow	<concept>	image
	aide_stat	élément d'information
	méthode	élément d'information
	hlp_guide	élément d'information
	exemple	élément d'information

Ces quatre derniers figurant principalement pour servir de défaut aux exercices.
autres attributs du lien:
cmd = D|L|...
temps = <entier>
statut= <entier>
exercice = <handle>| « »

Note: un élément de connaissance porte le nom de la base qui peut servir d'élément d'entrée lors de la création et de l'écriture de la base.

Objet: image
attribut: name = <nom>
attribut: printname = <string>

attribut: type = image
attribut: system = comp|...
(attribut: coordonnées)
content: codeimage(<string>)
linklist:

nom	handle	pointe sur
slideshow	<concept>	image

autres attributs du lien:
cmd = D|L|...
temps = <entier>
statut= <entier>

Objet: préparation slideshow (nécessaire lorsque par de link préalable)
attribut: name = <nom du slideshow>
attribut: type = relais
content: text(REF) description
linklist:

nom	handle	pointe sur
slideshow	<concept>	image

autres attributs du lien:
cmd = D|L|...
temps = <entier>
statut= <entier>

Objet: règle
attribut: name = <nom de la règle>
attribut: type = relais
content: text(REF) énoncé de la règle
linklist:

nom	handle	pointe sur
	CtrlGuide	chemin

Objet: chemin
attribut: name = nom de la règle + indice (àmettre en attribut?)
attribut: type = aiguillage
attribut: chemin = <chaîne listée>
content: nil

Objet: exercice
attribut: name = <nom de l'exercice>
attribut: printname = <titre de l'exercice>
attribut: contexte = <chaîne listée>
attribut: type = exercice
attribut: concept = <concept>
attribut: window = CX/CY/H/L/WI/FR
attribut: difficulté = INTEGER

content: text(consigne)
linklist:

nom	handle	pointe sur
	question	question
	aide_stat	élément d'information
	distracteur	donnée
	guide	règle
	<1er concept>	élément d'information
	donCmpl	donnée (formule(TRAITEMT,VALEURS)*
	exemple	élément d'information
	méthode	élément d'information
	hlp_guide	élément d'information
	msg	élément d'information

* valeurs pour l'énoncé, traitement complémentaire pour la réponse)

Objet: question
attribut: name = nom de la question *
attribut: contexte = <chaîne listée>
attribut: aremp =
attribut: remp =
attribut: type = question
attribut: concept =
attribut: window =
attribut: difficulté =
content: text(consigne) ; nil
linklist:

nom	handle	base pointée
	énoncé	donnée
	réponse	donnée
	aide_stat	élément d'information
	guide	règle
	<1er concept>	élément d'information
	exemple	élément d'information
	méthode	élément d'information
	hlp_guide	élément d'information

Objet: donnée
attribut: name
attribut: printname
attribut: type = donnée
attribut: window =
attribut: difficulté = (lecture)
content: don(DON)

DON = expr(string) ; expr2(string,liste,string) ;

exp(expression algébrique) ;
probleme(string,liste_variables, liste_valeurs) ;
formule(expression algébrique, liste_valeurs) ;
list(liste de données)

VALEUR = valeur(nom,expression algébrique)

Objet: élément de vocabulaire
attribut: name = mot
attribut: type =
content: text(REF) définition
linklist:

nom	handle	pointe sur
descripteur	<concept>	élément d'information

Annexe E: Description de quelques variables

L'UI ui_sys2x2 constitue la première aide pour la résolution d'un système de deux équations àdeux inconnues par la méthode d'addition. Les variables utilisées sont:

- dimX: quantité représentée par la variable x.
- dimY: quantité représentée par la variable y.
- a, s1, b1, u, c, s2, d1, v: coefficients et signes du système d'équations.
- suite_sys: la valeur de cette variable, calculée par l'expert de résolution de systèmes de 2 équations à deux inconnues contiendra, l'hyperchamp désignant l'opération à effectuer et le handle nécessaire pour passer àl'UI présentant la suite des opérations.

A noter que dans ce cas un certain nombre de link sont àpréparer, correspondants au différents handles pouvant apparaître: add, sous1_2, sous_2_1, amp1, amp2, amp1_2. Dans les unités appelées, certains de ces liens sont aussi utilisés (par exemple àpartir d'un système amplifié, on peut passer àune addition). Mais il n'est pas nécessaire de les réaliser, l'héritage s'en charge !

Les UI appelées par cette première unité (et qui peuvent s'appeler entre elles) sont: ui_sys_amp1 (variables: par, equa1, equa2, suite_sys_amp), ui_sys_amp2 (variables: equa1, par, equa2, suite_sys_amp), ui_sys_amp1_2 (variables: par1, equa1, par2,equa2, suite_sys_amp), ui_sys_add (variables: equa_add, suite_sys_add), ui_sys_sous1_2 (variables: equa_add, suite_sys_add), ui_sys_sous_2_1 (variables: equa_add, suite_sys_add), ui_sys_sol (variables: sol1, no_eq, sous_sys, sol2).

L'UI ui_sys2x2_gen propose un choix entre une résolution par addition ou par substitution. Les variables sont les mêmes que celles de ui_sys2x2 avec en plus: eq1_gen

L'UI ui_tiroir1 propose une transformation menant àune équation (1 inconnue). Les variables sont: x1, a1, a, c1, c2, c, x, dimX, eq, sous-but, nouvelle-donnée.

L'UI ui_tiroir2x2 propose une transformation menant àun système de deux équations à deux inconnues. Variables: celles de ui_sys2x2 avec en plus: t1, t2, t, sous-but, nouvelle-donnée.

L'UI ui_prop constitue la première aide pour la résolution de problèmes de proportionnalité. Les variables utilisées sont:

- 1ère suite, 2ème suite: en-tête des lignes.
- X1, X2: valeurs sur la première ligne.
- Y1, Y2: valeurs sur la deuxième ligne.