Principes de conception d'un système pour enseigner la résolution des problèmes par la modélisation

Design principles for a system to teach problem solving by modelling.

Gérard TISSEAU,
Hélène GIROIRE
Equipe SysDeF - LIP6
Université Paris6
Boîte 169, Tour 46-0 2° étage
4 Place Jussieu
75252 Paris Cedex 05
France
E-mail : <Prenom>.<Nom>@lip6.fr
Phone (33) 1 44 27 70 05
Fax (33) 1 44 27 88 12

Françoise LE CALVEZ,
Marie URTASUN
CRIP5
Université René Descartes
45 rue des Saints Pères
F-75270 Paris Cedex 06
France
Francoise.Le-calvez@math-info.univ-paris5.fr
Phone : (33) 1 44 55 35 47
Fax : (33) 1 44 55 35 35

Jacques DUMA
Lycée technique Jacquard
2 rue Bouret
75019 Paris
France
dumajd@club-internet.fr

Résumé

Cet article présente une approche pour réaliser un environnement d'aide à l'apprentissage humain dans un domaine mathématique (les dénombrements) où la résolution de problèmes repose plus sur la modélisation que sur la déduction ou le calcul. Dans cette approche, nous avons voulu offrir à l'étudiant une présentation proche des formulations usuelles en langue naturelle qu'il a tendance à donner spontanément, tout en garantissant la rigueur mathématique des raisonnements. Pour cela, nous avons introduit trois niveaux de modélisation : d'abord une formalisation mathématique de la démarche intuitive des élèves, ensuite un modèle conceptuel informatique permettant aussi bien d'effectuer des raisonnements mathématiques que de communiquer avec l'élève, et enfin une présentation suivant plusieurs interfaces regroupant chacune une classe différente de problèmes. Ces interfaces sont vues comme des "machines" nondéterministes que l'élève utilise pour construire une configuration répondant à des contraintes.

Mots Clef

Tuteurs intelligents, aide à l'apprenant, interaction homme-machine, représentation de connaissances, dénombrement.

Abstract

This paper presents an approach to the design of a learning environment in a mathematical domain (elementary combinatorics) where problem solving is based more on modelling than on deduction or calculation. In this approach, we want to provide the student with a presentation which is close to the natural language formulations that she tends to give spontaneously, while ensuring a rigorous mathematical reasoning. To do so, we have introduced three modelling levels : first a mathematical formalisation of the students' intuitive process, then a conceptual and computational model allowing mathematical reasoning as well as communication with the student, and finally a presentation consisting in several interfaces, each one grouping problems ot some class. These interfaces are viewed as nondeterministic "machines" that the student uses to build a configuration satisfying some constraints.

Keywords

Intelligent tutoring system, Knowledge representation for instruction, interface design, combinatorics.

1. Introduction

Un "système tutoriel intelligent" propose à l'apprenant des activités lui permettant d'acquérir des connaissances ou des savoir-faire dans un domaine d'étude. Classiquement, ces activités sont centrées sur la résolution de problème : le système propose un problème à résoudre et l'apprenant doit fournir la solution. Le système examine alors la proposition de solution, l'évalue et la commente. Bien que ce schéma soit général, il peut s'instancier en des systèmes assez divers, en particulier en fonction de la forme des solutions, du type de démarche nécessaire à l'élaboration de cette solution, et des ressources offertes à l'apprenant pour l'aider à effectuer sa recherche et à exprimer sa solution.

Dans le domaine mathématique, les problèmes proposés sont souvent solubles par une méthode *déductive* : en partant de l'énoncé considéré comme un ensemble d'axiomes, l'apprenant peut appliquer des règles d'inférence pour obtenir de nouveaux faits et construire ainsi progressivement un chemin menant à la propriété à démontrer ou au résultat du calcul demandé. Pour évaluer l'activité de l'apprenant, le système peut observer quelles règles sont essayées, si elles sont correctement appliquées, si l'apprenant a bien atteint le but demandé et si le chemin trouvé est de bonne qualité. Au cours de la recherche, le système peut guider l'apprenant à chaque étape, par exemple en donnant des indications sur la pertinence des

nouveaux faits obtenus, en détectant des impasses, en essayant de deviner le plan suivi et en proposant des sous-buts.

Alors que ce type de démarche est privilégié dans des domaines comme la transformation d'expressions algébriques et la démonstration de propriétés géométriques [12], il n'est pas nécessairement adapté à tous les domaines. Dans certains domaines, comme les dénombrements, cette inadaptation se manifeste par les réactions des élèves face aux problèmes : "je n'arrive pas à me représenter le problème", "je ne sais pas par où commencer", "je comprends la solution donnée par le professeur mais je ne comprends pas pourquoi ma solution est fausse", "je propose une solution mais je n'ai pas de critères décisifs me permettant de savoir si elle est correcte ou non". Nous interprétons ces réactions comme des difficultés de *représentation* : l'essentiel de la résolution ne provient pas d'un enchaînement habile d'inférences ou de calculs, mais de l'établissement d'une représentation appropriée et de la transformation d'une représentation en une autre, équivalente et adaptée à l'exploitation des connaissances. L'importance des représentations dans la résolution de problèmes est étudiée par exemple dans [1, 4, 16].

Nous nous intéressons plus particulièrement à ces domaines car ils sont un bon support pour acquérir des capacités importantes, comme celle d'élaborer des modèles. Mais du fait des difficultés qu'ils présentent et des démarches particulières qu'ils nécessitent, ils restent un peu à part dans les programmes d'enseignement, qui n'exploitent pas vraiment leurs potentialités. Dans l'enseignement général en France, par exemple, les dénombrements sont traités rapidement et presque uniquement comme outil pour les probabilités. Il y a donc là un manque, et nous pensons qu'une approche de type "Système Tutoriel Intelligent" peut le combler.

Cet article présente une approche pour réaliser un environnement d'apprentissage dans le domaine des dénombrements au niveau de la classe de terminale, en prenant en compte les particularités de ce type de domaine. Nous présentons d'abord le projet COMBIEN? dans lequel cette approche s'insère, puis nous décrivons les activités proposées à l'apprenant et le mode d'interaction associé : les interfaces sont vues comme des "machines" nondéterministes que l'élève utilise pour construire une configuration répondant à des contraintes. Nous discutons ensuite les choix de modélisation que nous avons faits pour offrir à l'étudiant une présentation proche des formulations usuelles en langue naturelle qu'il a tendance à donner spontanément, tout en garantissant la rigueur mathématique des raisonnements.

2. Le projet COMBIEN?

Notre projet (appelé "projet COMBIEN?") est de réaliser un environnement interactif d'apprentissage permettant à un apprenant de se familiariser avec les notions mathématiques nécessaires à la résolution rigoureuse des problèmes de dénombrement.

2.1 Présentation et historique du projet

Depuis le début du projet COMBIEN?, nous avons mené de front des réflexions théoriques et des expériences d'implémentation. A partir de notre expérience dans l'enseignement des dénombrements en classe, nous avons défini les fondements mathématiques d'une méthode de résolution (la méthode constructive) adaptée aux conceptions usuelles des élèves et permettant d'accéder à la théorie mathématique du domaine [11] [14]. Nous avons défini une classification des problèmes du domaine et les schémas de résolution associés aux différentes classes. Cette classification a servi de base à un système d'IA qui résout des problèmes de dénombrement [8] [9] [10]. Nous avons défini un langage permettant de représenter un problème et une solution suivant la méthode constructive, et nous avons implémenté un système d'IA capable de vérifier qu'une solution écrite dans ce langage vérifiait bien le principe de base des dénombrements. Ce langage s'étant révélé trop général et abstrait, nous avons ensuite défini un modèle conceptuel objet du domaine mieux adapté à une utilisation interactive. Ce modèle a donné lieu à différentes implémentations expérimentales d'interfaces de saisie [2]. Parallèlement, nous avons réalisé différents prototypes d'interfaces pour explorer diverses possibilités d'interaction [6]. Nous avons alors retenu un certain type d'interface que nous avons appelé "machines à construire des configurations" [11].

L'article expose les principes de conception qui sont à la base de ces machines.

2.2 Un domaine difficile : les dénombrements

Voici un exemple de problème de dénombrement : avec un jeu de 32 cartes, combien peut-on former de mains de 5 cartes comportant 2 piques et 2 coeurs ? Les problèmes de dénombrement que nous considérons ont tous une forme analogue : étant donnés des ensembles servant de *référentiels* (ici le jeu de cartes), compter dans un certain *univers* (ici l'ensemble de toutes les mains de 5 cartes possibles) les éléments vérifiant des *contraintes de sélection* (ici, comporter deux piques et deux coeurs). Résoudre rigoureusement ce type de problème demande d'introduire une représentation appropriée, souvent abstraite et complexe, utilisant les concepts de la théorie des ensembles (par exemple ensembles, applications, ensembles d'ensembles). Pour un débutant, cette approche est inaccessible car la représentation est très éloignée de l'image mentale concrète.

Pourtant, certains élèves sont capables de résoudre ces problèmes (voir [7] pour une étude de la capacité de dénombrement des enfants et des adolescents). Par exemple, si on leur demande "combien y a-t-il de mots de cinq lettres comportant exactement deux occurrences de la lettre A ?", leur réponse serait du type suivant : "je choisis d'abord les deux places pour A, ce qui me donne 10 possibilités, puis je complète les places restantes, ce qui me donne 25×25×25 possibilités (15625). Il y a donc en tout 156250 mots répondant à la question". Une telle réponse ne constitue cependant pas une démonstration mathématique et les élèves ne sont généralement pas

capables de justifier tous les aspects de leur réponse. De plus, la simplicité apparente de cette réponse cache la difficulté que certains ont à la trouver : beaucoup d'élèves n'arrivent pas à résoudre correctement le problème.

2.3. Objectifs pédagogiques

Notre objectif est d'introduire quelques notions de base des dénombrements fondées sur la théorie des ensembles (par exemple les ensembles, les listes, les applications d'un ensemble dans un autre, l'égalité d'ensembles définis par des contraintes) et quelques schémas de raisonnements qui s'y appliquent. L'important est de faire acquérir les représentations et les pratiques du domaine pour préparer un cours théorique ultérieur, de manière à rendre compréhensibles les théorèmes abstraits de la théorie formalisée. Pour arriver à ce but sans formalisation excessive, nous donnons à l'apprenant la possibilité de s'exprimer en des termes analogues à ceux de la réponse usuelle d'un élève donnée ci-dessus.

Pour mettre en oeuvre notre projet, nous avons d'abord effectué une étude théorique du domaine, pour bien préciser ce que nous voulions que l'élève apprenne et quelles activités pourraient lui permettre d'y arriver. Nous avons pour cela formalisé mathématiquement la méthode correspondant au type de réponse donnée habituellement par les élèves, méthode que nous avons appelée "constructive" [14] (en effet, elle repose sur un point de vue dynamique décrivant un enchaînement d'actions : "je choisis", "puis je complète"). Nous avons ensuite défini des activités à proposer à l'élève, reposant sur l'aspect dynamique de la méthode : puisque l'élève parle de lui-même de "choisir" et de "compléter", offrons lui des activités qui consistent à choisir et à compléter. Partons de ce que l'élève sait faire pour l'amener progressivement à une prise de conscience des structures abstraites sous-jacentes et des méthodes rigoureuses de raisonnement mathématique qui s'appliquent à ces structures.

L'objectif n'est pas tellement de former des experts en dénombrement, c'est-à-dire capables de déterminer le nombre d'éléments d'un ensemble, mais plutôt d'entraîner les élèves à la modélisation et de les rendre capables de représenter une situation par une structure complexe. Cette capacité est essentielle dans les activités de conception, comme l'atteste l'importance croissante accordée actuellement aux méthodes de modélisation en génie logiciel. Nous pensons que les problèmes de dénombrement sont un bon point de départ pour cela et que la même démarche se retrouve dans d'autres domaines comme les probabilités et l'algorithmique. De plus, la résolution de problèmes de dénombrements demande non seulement de savoir décrire une structure, mais aussi de raisonner sur une telle description. Par exemple, on pourra décrire un ensemble comme étant égal au produit cartésien de l'intervalle [1, 100] par l'intervalle [4, 33] et on pourra compter le nombre de ses éléments ($100 \times (33 - 4 + 1)$, soit 3000) sans jamais énumérer la totalité de ces éléments. La taille des ensembles mis en jeu nécessite un tel raisonnement et rend plus convaincant le recours à l'abstraction.

3. Les activités de l'élève

Les activités que nous proposons à l'apprenant ne consistent pas à dénombrer mais à décrire, car nous voulons insister sur les problèmes de représentation. Avant de se poser la question "Combien ?", il faut d'abord se poser la question "Quoi ?", c'est-à-dire donner une représentation précise des configurations que l'on cherche à dénombrer. Au cours d'une même activité, l'élève doit répondre à trois exigences : il doit produire un résultat, il doit utiliser obligatoirement certaines ressources pour produire ce résultat et la démarche qu'il suit doit être générique. Nous détaillons ces éléments dans ce qui suit. En ce qui concerne le rôle des exemples dans l'apprentissage des mathématiques voir [13].

Le *résultat* que doit produire l'élève à la fin de chaque activité est un exemple de configuration répondant à des contraintes. Par exemple, construire un mot de cinq lettres comportant deux occurrences de la lettre A. Une réponse possible est le mot "ALPHA". Mais, pour lui faire prendre conscience de la structure d'une telle configuration, nous ne lui demandons pas d'écrire simplement un exemple, nous l'obligeons à utiliser certaines ressources mises à sa disposition.

Les *ressources* offertes à l'apprenant se présentent sous la forme d'une "machine" à laquelle l'élève doit donner des instructions pour qu'elle produise le résultat demandé. Nous avons choisi le jeu d'instructions et les contraintes de fonctionnement d'une telle machine pour amener l'apprenant à prendre conscience des représentations et des principes sous-jacents. Une instruction est par exemple : "choisir 3 places et les remplir avec des lettres choisies au hasard dans l'ensemble des lettres différentes de A". La machine exécute cette instruction en disant : je choisis les places 2, 3 et 4. A la place 2, je mets un L, à la place 3 je mets un P et à la place 4 je mets un H. Cette présentation suggère la notion mathématique d'application de l'ensemble des places dans l'ensemble des lettres. Pour résoudre le problème précédent, l'apprenant doit suivre à peu près la démarche suivante :

1. Je déclare que je vais utiliser 5 places numérotées de 1 à 5.
2. Je déclare que je vais utiliser l'alphabet usuel à 26 lettres.
3. Au départ, toutes les places sont libres.
4. Je veux que la machine choisisse d'abord 2 places libres et les remplisse avec la lettre A.
5. Je veux que la machine choisisse ensuite 3 places libres et les remplisse avec des lettres tirées au hasard dans l'ensemble des lettres différentes de A, avec répétitions autorisées.
6. J'aurai alors terminé : j'aurai construit un mot répondant à la question.

L'activité de l'apprenant est donc une sorte de programmation, avec des déclarations et des instructions. Mais, contrairement à la programmation classique, les programmes peuvent contenir ici des instructions non déterministes. Ce n'est pas l'élève qui choisit les places

ou les lettres, c'est la machine qui le fait aléatoirement, en respectant les contraintes imposées par l'élève. L'effet d'une telle instruction est la plupart du temps imprévisible, du moins en ce qui concerne les éléments précis choisis.

La *démarche* de l'élève doit être générique au sens suivant : il doit écrire des instructions telles que, si la machine les exécutait de toutes les façons possibles, elle obtiendrait toutes les configurations solutions, rien que des configurations solutions, et chaque configuration seulement une fois. Cette règle est ce que nous appellerons dans la suite le *principe de base des dénombrements* : n'oublier aucune solution, ne compter que des solutions et ne rien compter plusieurs fois. Ce principe est lié au théorème mathématique suivant : deux ensembles ont même cardinal si et seulement si il existe une bijection de l'un sur l'autre. C'est bien pour faire assimiler ce principe sous ses différents aspects et en faire reconnaître l'importance que nous avons conçu ce type d'activité. L'apprenant doit donc formuler ses requêtes de manière générique et cette obligation l'amène à décrire la configuration demandée en termes abstraits, dans le cadre fixé par la machine. Pour rendre cette abstraction acceptable, nous avons choisi de ne pas lui demander d'écrire d'abord un programme complet et de l'exécuter ensuite. Au contraire, chaque instruction est exécutée immédiatement, de manière à lui permettre de voir où il en est et ce qui lui reste à faire. La machine doit donc jouer plutôt le rôle d'un interpréteur que d'un compilateur.

4. Interface et interaction

Une machine se présente dans une fenêtre comme une sorte de formulaire contenant des éléments d'interactions (widgets) de type classique (menus, boutons, champs). Chaque machine concerne une classe particulière de problème, mais toutes les machines ont le même scénario d'utilisation : l'apprenant choisit un exercice à résoudre, il en fournit un modèle puis il donne une construction d'une configuration solution. Les énoncés d'exercices proposés par une machine sont mémorisés de manière prédéfinie dans cette machine et chacun est associé à une représentation interne du problème.

L'élève choisit un énoncé en langue naturelle par l'intermédiaire d'un dispositif de sélection (un menu associé à une case de texte) : voir la Figure 1.

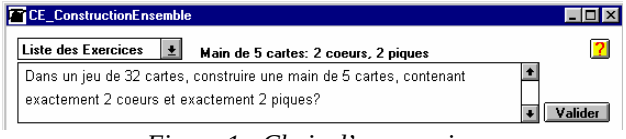

Figure 1 : Choix d'un exercice

Il doit ensuite préciser certains éléments permettant de modéliser le problème. Par exemple, pour les problèmes du type "construire une main de 5 cartes avec 2 piques et 2 coeurs", il doit préciser combien il veut d'éléments et dans quel ensemble la machine doit les prendre, en lui laissant le choix parmi des réponses prédéfinies, présentées dans un menu (voir la figure 2). Dans la théorie sous-jacente, ces informations définissent « l'univers ».

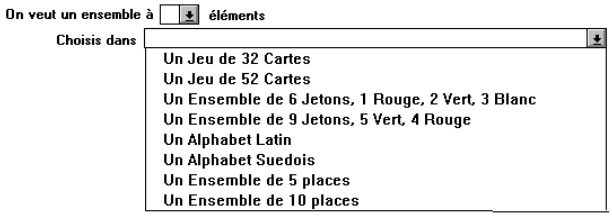

Figure 2 : Définition d'un univers

Pour l'apprenant, chaque réponse prédéfinie renvoie à son expérience usuelle et cela suffit dans un premier temps. Le système, lui, associe à chaque réponse un objet interne entièrement modélisé. Cette première phase de modélisation de l'énoncé reste ainsi très brève, tout en permettant une prise de conscience des choix de représentation. Cela équivaut à une sorte de contrat : en précisant cela, l'élève exprime lui-même ce qu'il s'engage à réaliser. De plus, cela permet à la machine de ne présenter dans la suite que des possibilités correspondant à ce choix. On voit aussi sur cet exemple en quoi la spécialisation des machines évite à l'apprenant de choisir tous les aspects de la modélisation : ici, c'est la machine elle-même qui indique qu'on construit des "ensembles". Une machine générale aurait pu demander le type de configuration ("ensembles", "listes", "applications", etc.).

L'apprenant décrit ensuite sa construction sous forme d'une suite d'instructions toutes du même type, appelées contraintes. La figure 3 montre un exemple de saisie d'une contrainte et la figure 4 montre la récapitulation des contraintes déjà saisies. Sur la figure 4, une des contraintes est sélectionnée, et des boutons permettent de la supprimer ou de la modifier.

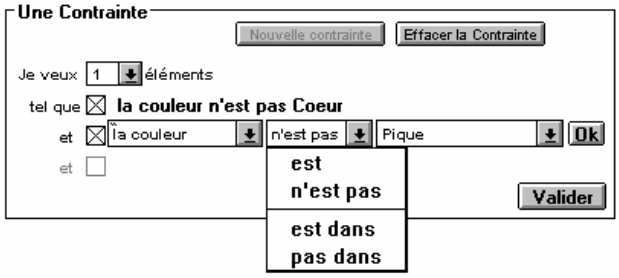

Figure 3 : Saisie d'une contrainte

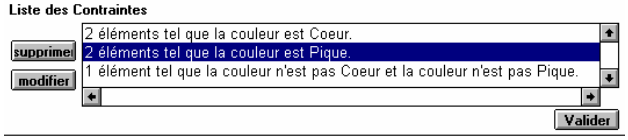

Figure 4 :Liste des contraintes

La présentation est celle d'un texte écrit en langue naturelle, comportant des espaces à compléter. La syntaxe des phrases et le type de mots proposés permettent de comprendre le rôle de chacun des éléments sans qu'il soit nécessaire de les étiqueter par des concepts abstraits, comme "attribut", "comparateur", "valeur". Le contenu des menus est dynamique : le dernier menu (les valeurs

possibles) a été mis à jour lorsque l'élève a choisi l'attribut "la couleur". L'apprenant demande ensuite à la machine d'exécuter l'instruction, ce qui fournit un ensemble d'éléments (par exemple un 7 de pique et une dame de pique si la contrainte est : « 2 éléments dont la couleur est pique ») qui sont ajoutés automatiquement à la configuration en cours, qui reste toujours visible à l'écran. Cet ajout automatique rend implicite la dernière instruction de la construction, qui serait : calculer la réunion de tous les morceaux engendrés. La configuration finale pourrait être par exemple :

{7 de pique, dame de pique, 10 de coeur, 8 de coeur, roi de carreau}

La machine contrôle le déroulement du dialogue de plusieurs façons. Elle réagit aux choix de l'apprenant en rendant disponibles ou indisponibles certains widgets de la fenêtre et en changeant dynamiquement le contenu des menus. Le but est de restreindre le nombre de degrés de liberté de l'apprenant pour le guider dans son parcours et lui éviter des erreurs dues à des incohérences sans intérêt pédagogique. Cependant, la machine lui laisse la possibilité de faire des erreurs concernant les concepts de dénombrement. Elle réagit en refusant d'exécuter l'instruction en cause. Parfois il s'agit simplement d'une impossibilité matérielle : "je ne peux pas choisir 3 piques, car il n'en reste plus que 2 dans le jeu". Mais les erreurs les plus intéressantes sont les violations du principe de base des dénombrements : "je refuse d'exécuter 'choisir 2 piques' car vous avez déjà demandé 'choisir 3 piques' dans une instruction précédente. Si j'acceptais, certaines configurations seraient engendrées plusieurs fois, comme par exemple …". La machine étant spécialisée dans un schéma de programme particulier, elle possède les connaissances nécessaires pour faire cette vérification. Par exemple, dans ce cas : si deux instructions demandent de choisir des éléments dans des ensembles E et F non disjoints, alors une configuration formée avec e dans E et f dans F sera comptée une autre fois avec f dans E et e dans F si e et f sont communs à E et F.

5. Modélisation

Les interfaces précédentes sont fondées sur une représentation interne résultant d'une modélisation du domaine et des exercices. Nous discutons ici les problèmes de modélisation qui se posent, les choix que nous avons retenus et les leçons qu'on peut en tirer.

5.1. Le choix d'un langage pour les programmes de l'apprenant

Comme l'activité proposée à l'apprenant est une forme de programmation, il faut définir les concepts utilisés dans ce type de programmation et un langage associé. Nous avons d'abord défini un langage fondé sur les notions de base de la théorie des ensembles, comme par exemple énumération, intervalle, produit cartésien, réunion, complémentaire, partie d'un ensemble, combinaison, fonction, injection, application réciproque, singleton. Pour décrire la construction d'un mot de cinq lettres comportant deux A, on avait le programme suivant :

Soit E1 l'ensemble des lettres A à Z
Soit E2 l'ensemble des nombres de 1 à 5
Soit E3 le sous-ensemble de E1 formé de la lettre A
Soit C3 le complémentaire de E3 dans E1
Soit E4 une partie à 2 éléments de l'ensemble E2
Soit C4 le complémentaire de E4 dans E2
Soit E5 une application de E4 dans E3
Soit E6 une application de C4 dans C3
Le résultat R est la réunion de E5 et E6

Une façon d'effectuer ces instructions fournit par exemple E4 = {1, 5}, E5 = {(1, A), (5, A)}, E6 = {(2, L), (3, P), (4, H)} et le résultat correspond alors au mot "ALPHA". Nous avons réalisé un prototype utilisant ce langage, mais il est apparu qu'il ne pourrait pas être proposé à un élève. Bien que chaque instruction soit élémentaire, l'arrangement des instructions pour obtenir le résultat n'est pas facile à comprendre et encore moins à inventer. Un exemple d'exécution peut aider à comprendre, en particulier pour les instructions non déterministes, mais cela ne dit pas comment on peut inventer une telle suite d'instructions. Le langage n'aide pas à le voir parce qu'il ne possède pas de construction structurée : la seule structure apparente est une séquence d'instructions. On se retrouve alors à peu près dans la même situation qu'avec un langage d'assemblage : puissant et général, mais peu structuré et à un niveau de granularité trop bas. Il est probable que ce phénomène se produise dans la plupart des domaines où la résolution de problèmes repose sur une phase importante de modélisation : même s'il existe une théorie mathématique générale qui permet de représenter les problèmes, elle risque de ne pas être adaptée comme moyen de communication avec l'étudiant. Nous avons alors abandonné cette approche fondée sur un langage général pour introduire le concept de schéma de programme, que nous présentons dans ce qui suit.

5.2. Les classes de problème et les schémas de programme

En fait, le programme précédent a une structure : il comporte des déclarations d'ensembles fixes (E1, E2, E3, C3) nécessaires à la construction, puis des instructions non déterministes (E5, E6) servant à construire des parties du résultat et utilisant des instructions auxiliaires (E4, C4), et enfin la formation du résultat R comme réunion des parties. De plus cette structure découle naturellement d'un plan selon lequel on peut construire un mot (R) comme réunion d'applications (E5 et E6).

Mais demander à un débutant d'inventer une telle structure et un tel plan à partir de rien est trop exigeant. Pour un expert, cette structure et ce plan correspondent à un schéma classique de résolution associé à une certaine classe de problème : construire un mot de taille donnée et comportant x_1 occurrences de lettres prises dans une certaine catégorie c_1, x_2 occurrences dans une catégorie c_2, etc. Un expert connaît différentes classes et leurs schémas de résolution associés, et cela l'aide à résoudre les problèmes classiques : il analyse d'abord le problème pour reconnaître sa classe, puis il applique un schéma de résolution associé. En fait utiliser une classification pour résoudre des problèmes est une méthode générale. [5]

propose une classification pour les configurations combinatoires simples mais elle ne recouvre pas exactement les problèmes que nous considérons et elle n'a pas été conçue pour être utilisée dans un environnement d'apprentissage. Nous avons défini une classification des problèmes du domaine permettant de résoudre la plus grande partie des problèmes posés au niveau considéré. Cette classification a servi de base à un système automatisé de résolution de problèmes [9, 10], ce qui a permis d'une part de montrer qu'elle était opérationnelle et d'autre part de montrer que la tâche de classification nécessitait une expertise non négligeable.

Reconnaître une classe de problème et appliquer un schéma de résolution sont des tâches complexes. Par exemple, il n'est pas immédiat de reconnaître que le problème suivant : "dans un groupe de 10 personnes, combien y a-t-il de façons de choisir un président, un trésorier et deux secrétaires ?" peut être vu comme une instance de la classe : "construire une fonction d'un ensemble A vers un ensemble B, comportant exactement n couples, avec x_1 couples ayant leur deuxième composante dans la catégorie b_1, x_2 couples ayant leur deuxième composante dans la catégorie b_2, etc., les catégories étant disjointes". Le problème initial n'est en effet pas posé dans ces termes abstraits, et il faut d'abord le modéliser avant de reconnaître sa classe, ce qui suppose parfois de faire des hypothèses qui ne sont pas explicitées dans le texte. Ici, par exemple, le problème n'est de la classe considérée que si une même personne ne peut pas avoir plusieurs rôles (le président ne peut pas être aussi trésorier).

Dans l'interface proposée, l'existence des classes est suggérée simplement par l'existence de regroupements de problèmes matérialisés à l'écran par différentes fenêtres, correspondant à différentes *machines à construire des configurations*. A chaque machine sont associés une classe et une méthode de résolution (sous forme d'un schéma de programme permettant de construire une configuration). Chaque machine propose des exercices qui peuvent être modélisés selon la classe correspondante et résolus par la méthode associée. L'idée importante est que cette classe et cette méthode sont inscrits dans l'apparence et le fonctionnement même de la machine.

Par exemple, pour la classe de problèmes du type : "Combien peut-on former de mains de 5 cartes dans un jeu de 32 cartes avec exactement 2 piques et exactement 2 coeurs ?", la machine comporte des widgets permettant à l'élève d'introduire un programme ayant le schéma suivant (l'aspect visuel a été montré dans la section 4) :

Je déclare que je vais prendre des éléments dans l'ensemble ⟨*une description d'ensemble*⟩

Au départ, tous les éléments sont disponibles.

Je veux que la machine choisisse ⟨*un nombre*⟩ éléments disponibles parmi ceux qui vérifient la condition ⟨*une condition*⟩

Je veux que la machine choisisse ⟨*un nombre*⟩ éléments disponibles parmi ceux qui vérifient la condition ⟨*une condition*⟩

etc.

La structure du programme est donc prédéterminée : deux schémas de déclarations, puis une séquence d'instructions ayant chacune le même schéma. L'apprenant n'a pas à réinventer cette structure, mais il lui reste quand même des choix à faire pour atteindre son but. Sa tâche peut paraître simple, mais l'expérience des réactions des élèves dans une classe prouve que même avec un tel schéma les erreurs ou les difficultés sont fréquentes. Par exemple, l'apprenant sera tenté de se contenter de répéter l'énoncé en disant : je veux d'abord 2 piques, puis 2 coeurs. Sa construction sera alors fausse car il n'obtiendra qu'une main de 4 cartes et non pas de 5. Le but est justement de lui faire prendre conscience que la simplicité apparente de sa démarche intuitive cache en fait des difficultés qu'il ne pourra résoudre qu'en explicitant les concepts sous-jacents. Par ailleurs cet exemple est un des plus simples : certaines classes ont des schémas nettement plus compliqués.

On peut retenir de cette discussion l'idée, applicable à d'autres domaines, de répartir les problèmes selon plusieurs classes et de présenter chaque classe d'une façon adaptée, plutôt que d'utiliser un formalisme général unique.

5.3. Modèle conceptuel du domaine et des problèmes

Une machine doit construire des configurations en suivant les instructions de l'apprenant et raisonner sur ces instructions, par exemple pour vérifier que le programme respecte le principe de base des dénombrements, cité plus haut. Elle doit donc posséder un modèle informatique interne du domaine qui se prête au raisonnement mathématique. Sur ce point, nous sommes d'accord avec Nicaud [12] quand il insiste sur l'importance de modèles décrits explicitement au niveau de la connaissance (knowledge level) et sur la nécessité d'élaborer une théorie de la résolution des problèmes du domaine. Malgré la diversité prévue des machines (une quinzaine), ce modèle doit être le même pour toutes les machines, d'abord pour faciliter la programmation et surtout pour refléter la cohérence interne du domaine. Il doit permettre de représenter non seulement des concepts mathématiques, mais aussi d'une part les problèmes posés et d'autre part l'activité de l'élève, qui est centrée autour d'une méthode particulière (la "méthode constructive" que nous avons définie) [14].

Nous avons défini un tel modèle sous forme d'un modèle conceptuel objet au sens des méthodes de génie logiciel [3], c'est-à-dire en définissant des classes d'objets munis d'attributs et des relations entre les classes. Les classes de plus haut niveau reflètent l'activité de l'apprenant : on lui donne un exercice sous forme de texte, et il doit en donner une solution, qui consiste en un modèle du problème et une construction des configurations demandées. Chaque machine concerne un type de construction associé à un type de problème.

La définition du modèle conceptuel est fondée non seulement sur la structure logique des principaux concepts

(problème, construction), mais aussi sur le souci de fournir à l'élève une représentation proche de son langage usuel. Cela a été rendu possible par le fait que la méthode théorique proposée (la "méthode constructive") avait elle même été élaborée à partir des réponses courantes des élèves.

Pour compenser la trop grande généralité du langage mathématique, où toutes les structures peuvent être assimilées à des ensembles, nous avons assigné des *rôles* différents aux structures intervenant en dénombrement, et plus particulièrement à celles qui interviennent dans la méthode constructive. Par exemple, les ensembles d'objets utilisés pour former des configurations ont été appelés des "référentiels" (un jeu de cartes, les lettres de l'alphabet, des jetons) et l'ensemble des configurations dans lequel on cherche les solutions a été appelé "univers". Ce vocabulaire et ces concepts ne figurent pas, à notre connaissance, dans les cours usuels sur le dénombrement, mais ils se sont révélés utiles. Certains ont été inspirés par d'autres domaines (comme "univers", emprunté aux probabilités). La figure 5 montre les classes liées au concept de référentiel. Un référentiel est une réunion de sous-référentiels, chacun étant défini par un ensemble de couples attribut-domaine. Par exemple, l'attribut « couleur » (pour un jeu de cartes), admet comme domaine les valeurs Pique, Cœur, Trèfle, Carreau. Un domaine peut être défini comme une énumération de valeurs littérales ou un intervalle de valeurs numériques ou alphabétiques

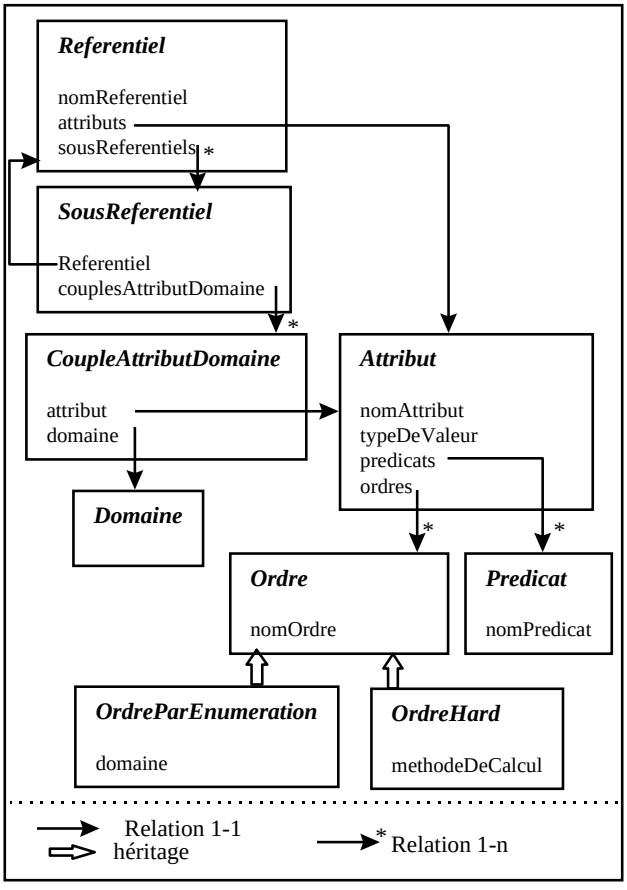

Figure 5 : Référentiel

A chaque attribut peuvent être associés des prédicats pouvant s'appliquer aux valeurs (par exemple le prédicat « pair » pour les nombres entiers). La figure 6 montre les classes permettant de modéliser les prédicats et les domaines. Certains prédicats sont définis par des méthodes programmées (classe PredicatHard, dont un exemple est « être pair ») et d'autres par l'ensemble des valeurs qui satisfont le prédicat (classe PredicatAppartenance, dont un exemple est « être une voyelle », défini par l'ensemble des valeurs {a, e, i, o, u, y}).

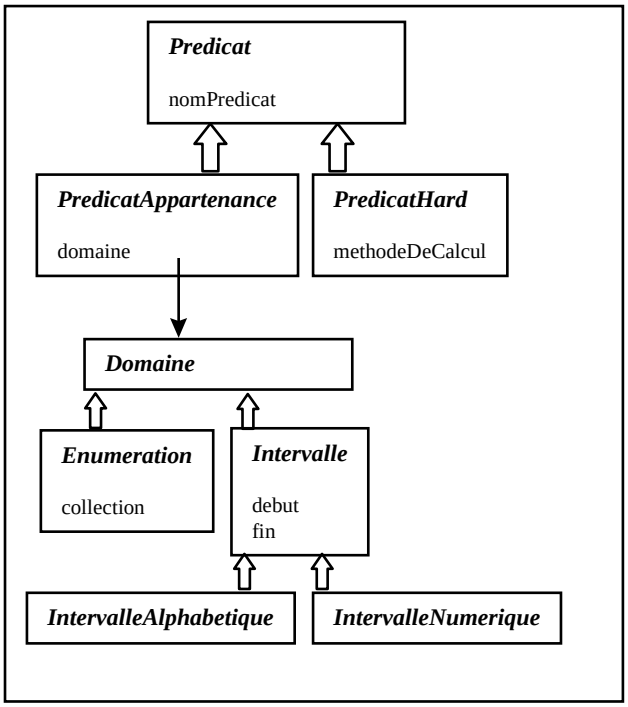

Figure 6 : Prédicats et domaines

Il y a deux grands types d'univers : UniversParties (pour représenter un ensemble de parties d'un référentiel, comme l'ensemble des mains de 5 cartes qu'on peut former avec un jeu de 32 cartes) et UniversAssociations (pour représenter un ensemble d'applications d'un ensemble dans un autre, comme l'ensemble des mots de 5 lettres, chaque mot étant vu comme une application de l'ensemble des 5 places dans l'ensemble des lettres). La figure 7 montre les concepts liés à la notion d'univers.

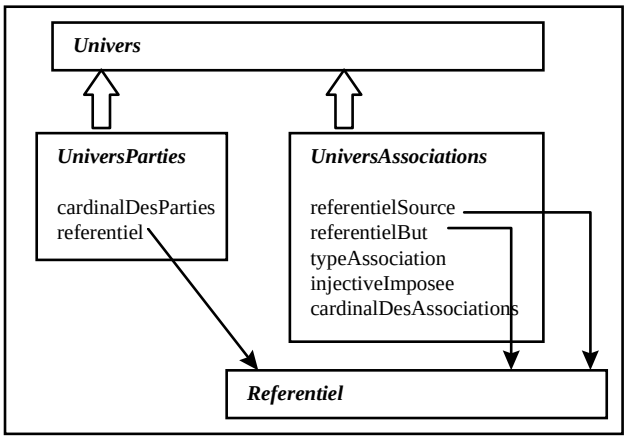

Figure 7 : Univers

Certains des concepts introduits ne concernent pas directement les objets mathématiques (ensembles, éléments, univers) mais les moyens d'expression offerts à l'apprenant. Cela introduit un niveau méta, que nous discutons dans ce qui suit.

5.4. Métamodélisation

Pour communiquer avec la machine, l'apprenant doit parler des objets contenus dans les référentiels, comme les cartes à jouer et les jetons (qui peuvent être numérotés et colorés). On dit souvent que les dénombrements ne nécessitent pas de connaître la nature ou les propriétés des objets manipulés, parce qu'on se préoccupe seulement de les compter. Ce n'est pas tout à fait vrai. Pour compter des configurations contenant des cartes à jouer, il faut bien savoir que chaque carte possède une couleur et une hauteur, et quelles sont les valeurs possibles de ces attributs. De même, il faut parfois savoir ce qu'est un nombre pair, ou ce que signifie qu'une lettre est une voyelle. Il faut donc que l'interface offre à l'apprenant des moyens d'expression pour faire référence à ces objets, ces attributs et ces propriétés, et que le modèle conceptuel contienne les concepts associés.

En fait, le seul aspect important est que les objets ont des attributs, que ces attributs ont une liste prédéfinie de valeurs possibles, et qu'on ne cherche pas à interpréter ou décomposer ces valeurs : elles sont uniquement atomiques. Cependant, ces valeurs peuvent posséder des propriétés unaires (un nombre peut être pair, une lettre peut être une voyelle) et peuvent être comparées entre elles. Nous nous sommes restreints à ces primitives de modélisation pour représenter les objets, nous rapprochant ainsi du modèle relationnel des bases de données. Les principaux concepts introduits ont été présentés dans figures 5 et 6.

De plus, nous avons retenu l'idée de définir un sous-ensemble d'un ensemble d'entités (comme un jeu de cartes) par une formule déclarative de *sélection.* L'ensemble des piques du jeu de cartes peut être désigné par "les cartes dont l'attribut couleur a la valeur pique". Cette notion de sélection permet de simplifier les constructions que doit écrire l'apprenant. Au lieu de définir préalablement les ensembles auxiliaires dont il aura besoin lors de sa construction (et de le faire par exemple à l'aide de produits cartésiens peu naturels), il peut les introduire seulement au moment où il en aura besoin, sous forme de sélection et sans les nommer, ce qui se rapproche d'une formulation en langue naturelle : "je veux 2 cartes dont la couleur est pique". La figure 3 a déjà montré comment l'élève pouvait introduire une telle formule de sélection. Dans le modèle, une telle formule est appelée un filtre. C'est une conjonction de propriétés, chacune pouvant être une comparaison (la couleur est pique), une appartenance (la couleur est dans {pique, carreau}) ou une assertion utilisant un prédicat (le numéro est pair). La figure 8 montre les principales classes nécessaires à la modélisation d'un filtre.

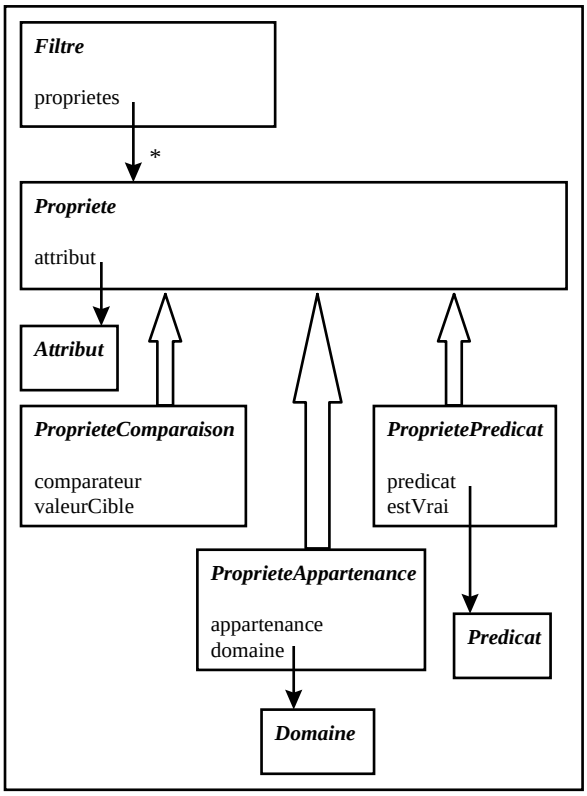

Figure 8 :Filtre

Finalement, notre choix consiste à introduire un niveau méta : au lieu de prévoir des classes prédéfinies d'objets (la classe des cartes à jouer, la classe des jetons, etc.), nous représentons explicitement un formalisme de représentation d'objets. Ce formalisme est concrétisé dans le modèle par les classes déjà présentées (Attribut, Predicat, Domaine, etc.). Un avantage de cette approche est de permettre à un enseignant (ou même à l'apprenant) d'introduire lui-même de nouveaux types d'objets sans avoir à modifier le modèle conceptuel ou à programmer. A travers l'interface, l'élève utilise ce formalisme pour exprimer ses solutions, mais la présentation cache les détails du formalisme et se rapproche d'un énoncé en langue naturelle, comme on l'a vu dans la section 4.

Ce formalisme sert également à représenter le problème. Pour chaque exercice, le système possède en effet, en plus de l'énoncé en langue naturelle, une représentation interne (classe Problème). Un problème est défini par un univers décrivant le type des configurations considérées (par exemple, l'ensemble des main de 5 cartes d'un jeu de 32 cartes) et un ensemble de contraintes, indiquant à quelles configurations de l'univers on s'intéresse (par exemple, les mains qui contiennent exactement 2 piques). La figure 9 montre les différentes classes de contraintes. Un exemple de ContrainteEffectif est « comporter exactement 2 éléments dont la couleur est pique ». Un exemple de ContrainteRestriction est « les lettres figurant aux places 1 et 3 doivent être des voyelles ». Un exemple de ContrainteDistribution est « comporter exactement 2 cartes d'une même hauteur et 3 cartes d'une autre hauteur ».

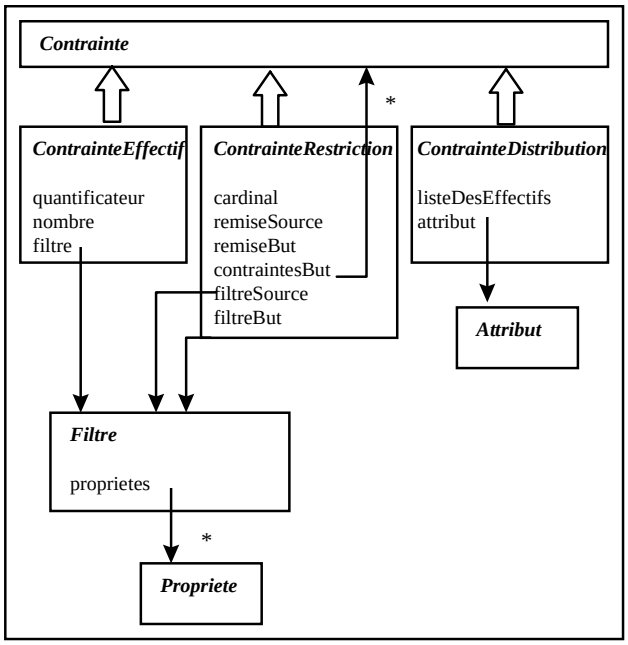

Figure 9 : Contrainte

5.5 Modèle des solutions

Dans la méthode constructive, la solution d'un problème est exprimée sous forme d'un programme de construction (classe Construction), tel qu'il a été défini en 5.1 et 5.2. Il y a deux grands types de construction, correspondant aux deux grands types d'univers : ConstructionDePartie et ConstructionDAssociation. Une construction de partie est par exemple : « choisir 2 cartes dont la couleur est pique, puis choisir 3 cartes dont la couleur est cœur ». Une construction d'association est par exemple : « choisir 2 places, les remplir avec la lettre a, puis remplir les autres places avec des lettres distinctes différentes de a ». La figure 10 montre les classes de construction.

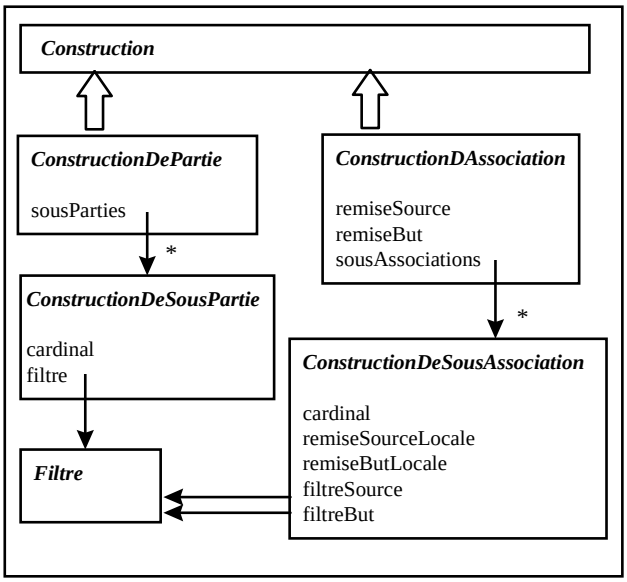

Figure 10 : constructions

Une construction est formée de sous-constructions, de telle sorte que la configuration finale (une main de 5 cartes) soit obtenue comme réunion des ensembles fournis par les sous-constructions (2 piques, 3 coeurs). La méthode constructive exige que les constructions vérifient certaines propriétés mathématiques pour être considérées comme valides. Par exemple, pour une construction de partie (choisir 2 piques puis 3 coeurs), les sous-ensembles dans lesquels on choisit (les piques, les coeurs) doivent être disjoints deux à deux.

Pour chaque exercice, le système possède un modèle complet de l'exercice (classe Problème) et une solution associée (classe Construction). Une session (classe Session) contient donc un exercice, avec son modèle et sa solution internes (classe Exercice) et une solution de l'élève (classe Solution). La figure 11 montre les classes correspondantes.

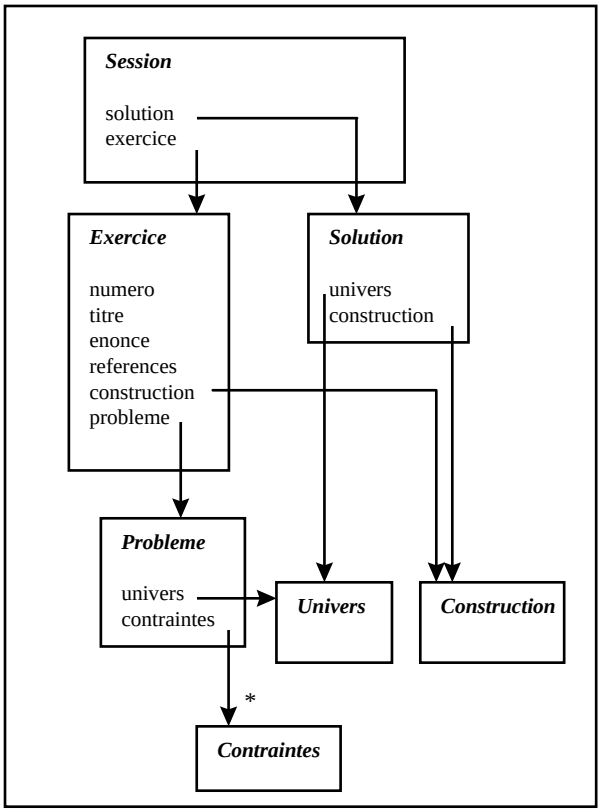

Figure 11 :Session

6. Conclusion

Cet article montre l'approche que nous proposons dans le projet COMBIEN? pour un environnement d'apprentissage sur les dénombrements, domaine mathématique dans lequel la résolution de problèmes présente des difficultés relevant plus de la modélisation et de la représentation que de la déduction ou du calcul. La réalisation d'un STI dans un tel domaine doit prendre en compte cette particularité, aussi bien en ce qui concerne les activités proposées à l'apprenant qu'en ce qui concerne le modèle conceptuel utilisé par le système et les types de présentation et d'interaction. Nous proposons des activités demandant à l'apprenant de modéliser le problème et d'effectuer une construction à l'aide de machines non déterministes, en évitant deux écueils : le premier, c'est que l'approche soit trop abstraite, avec une trop grande technicité mathématique, et le second, c'est qu'elle soit trop

concrète et ne puisse pas déboucher sur un réel apprentissage de notions mathématiques. Pour réduire l'abstraction, nous introduisons plusieurs machines associées à des classes de problèmes différentes et correspondant à des méthodes effectivement rencontrées chez les élèves. Pour relier les activités de l'élève à un raisonnement mathématique rigoureux, nous avons introduit deux niveaux conceptuels intermédiaires entre la théorie de base des dénombrements et les activités de l'élève : d'abord une méthode constructive de dénombrement et ensuite un modèle conceptuel utilisable à la fois dans les raisonnements et dans la présentation de l'interface.

Nous avons réalisé plusieurs prototypes pour tester ces idées et nous commençons à les intégrer dans un système opérationnel. Pour mettre en oeuvre plus facilement ces machines, nous avons défini un formalisme déclaratif pour spécifier ces interfaces [15] et nous avons réalisé un générateur de code qui engendre la plus grande partie de l'interface à partir de spécifications écrites dans ce formalisme. Notre prochain travail sera de spécifier et d'engendrer les différentes machines à l'aide de ce générateur, et de les tester avec des étudiants utilisateurs.

Bibliographie

[1] Abidin B., Hartley J.R., Developing mathematical problems solving skills, Journal of Computer Learning (1998) 14, 278-291.

[2] Annell M., Malthet V., Giroire H., Tisseau G., Construction d'interface de saisie d'un problème de dénombrement. Rapport interne Lip6 juin 99.

[3] Booch G., Rumbaugh J., and Jacobson I., Unified Modeling Language User Guide. Addison Wesley, (1998).

[4] Cox R., Brna P., Supporting the use of external representations in problem solving: the need for flexible Learning Environments. Journal of Artificial Intelligence in Education, 6, 2/3, 1995, p.239-302.

[5] Dubois J.-G., Une systématique des configurations combinatoires simples. Educational studies in Mathematics 15 (1984), p. 37-57.

[6] Duma J., Giroire H., Le Calvez F., Tisseau G., Urtasun M., Mise en évidence de styles de résolution, évolution de l'interface dans le projet COMBIEN ?, Actes des quatrièmes journées francophones EIAO, ENS de Cachan, 1995, EIAO, Tome 2, D. Guin, J.-F. Nicaud & D. Py (eds.), Eyrolles, 1995, 245-256.

[7] Fischbein E., Gazit A., The combinatorial solving capacity in children and adolescents. Zentralblatt für Didaktik der Mathematik, 5, 1988, 193-198.

[8] Guin N., Giroire H., Tisseau G., Le classement de problèmes : Une méthode de résolution pour le module expert d'un EIAO. Application au problème de dénombrement. Guin D., Nicaud J.F., Py D., (eds) Environnements Interactifs d'Apprentissage avec Ordinateur, Tome 2. Paris, Eyrolles, 1995, p. 113-124.

[9] Guin N., Changing the representation of a problem in order to solve it : use of classification, in proceedings of AI-ED 97, Kobe (Japan), August 18-22, 1997, p. 583-585.

[10] Guin N., Reformuler et classer un problème pour le résoudre. L'architecture SYRCLAD et son application à quatre domaines. Thèse de l'université Paris 6, décembre 1997.

[11] Le Calvez F., Urtasun M., Giroire H., Tisseau G., Duma J., Les machines à construire. Des modèles d'interaction pour apprendre une méthode constructive de dénombrement. EIAO'97, actes des cinquièmes journées EIAO de Cachan (Baron, M., Mendelsohn, P., Nicaud, J.-F., (eds), Hermès, 1997, p.49-60.

[12] Nicaud J.-F., Building ITSs to be used: Lessons Learned from the APLUSIX Project, Lessons From Learning, Lewis, R., Mendelsohn, P., (eds), IFIP, North Holland, 1994, p.181-198.

[13], Selden A., Selden J., The Role of Examples in Learning Mathematics, February 20, 1998, The Mathematical Association of America, URL http://www.maa.org/t_and_l/sampler/rs_5.html.

[14] Tisseau G., Giroire H., Le Calvez F., Urtasun M., Duma J., Une méthode "constructive" de résolution de problèmes de dénombrement et sa mise en oeuvre. Rapport interne Laforia 96/11, mai 1996.

[15] Tisseau G., Duma, Giroire H., Le Calvez F., Urtasun M., Les "machines à construire" : des interfaces dans le projet COMBIEN? Conception, spécification et réalisation. Actes du Colloque d'AI : Apprentissage et Acquisition de Connaissances, Berder 1998 Kornman S. . Rapport interne LIP6 98 n°7 p. 47-77.

[16] White B.Y., Frederiksen J.R., Causal model progressions for intelligent learning environments. Artificial Intelligence, 42, p.99-157.