# Génération de feuilles d'exercices de géométrie à l'aide d'énoncés indexés automatiquament

Mathieu Hibou, Jean-Marc Labat, Jean-Pierre Spagnol

▶ **To cite this version:**

Mathieu Hibou, Jean-Marc Labat, Jean-Pierre Spagnol. Génération de feuilles d'exercices de géométrie à l'aide d'énoncés indexés automatiquament. Environnements Informatiques pour l'Apprentissage Humain 2003, Apr 2003, Strasbourg, France. pp.247-258. edutice-00000141

**HAL Id: edutice-00000141**
**https://edutice.hal.science/edutice-00000141**

Submitted on 4 Nov 2003

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Génération de feuilles d'exercices de géométrie à l'aide d'énoncés indexés automatiquement

**Mathieu Hibou*, Jean-Marc Labat*, Jean-Pierre Spagnol**.**

**SAFE-CRIP5*
*45 rue des Saints-Pères 75270 Paris Cedex 06*
*mathieu.hibou@wanadoo.fr, Jean-Marc.Labat@math-info.univ-paris5.fr*
***SBC-CRIP5*
*jps0123@club-internet.fr*

RÉSUMÉ : *Pour générer automatiquement une feuille d'exercices portant sur des connaissances données, il faut que ces exercices soient indexés par les connaissances utiles pour les résoudre. L'idée de ce projet est d'utiliser un démonstrateur automatique de théorèmes, Argos, afin de réaliser une indexation d'exercices de géométrie à partir des théorèmes et propriétés qui servent à leurs résolutions.*

*Argos a été développé non seulement dans le but de résoudre des exercices, mais également de manière à pouvoir en rédiger des démonstrations comme le ferait un enseignant ou des élèves de collège ou de lycée. Il s'agit donc d'extraire les informations qui sont utiles pour notre indexation des preuves obtenues par le démonstrateur.*

*Pendant la résolution d'un exercice, Argos génère automatiquement des règles à partir des propriétés qui lui ont été fournies de manière déclarative. Nous avons donc créé un lien entre ces règles et les propriétés afin de pouvoir raisonner au niveau des propriétés de la géométrie et non des règles d'Argos qui sont d'une granularité trop fine. Nous avons catégorisé ces propriétés à l'aide d'une ontologie des théorèmes de géométrie au niveau de l'enseignement secondaire (environ 75 théorèmes pour catégoriser 150 propriétés d'Argos). Les propriétés utilisées par un exercice sont stockées dans une base de données. Il est alors facile de construire une feuille d'exercices sur un sujet donné en interrogeant la base de données. Ainsi le système d'aide que nous proposons permet une assistance au curriculum ou à la planification d'activités dans un EIAH.*

MOTS-CLÉS : *Ontologie, Métadonnées, Démonstration Automatique, Représentation des connaissances.*

## 1 Introduction et Problématique

Les collèges et lycées accueillent un public de plus en plus large et diversifié et dans le même temps l'attitude face au travail a également changé, ce qui a obligé les enseignants à changer leurs méthodes d'enseignement. En particulier, le manque d'homogénéité des classes du point de vue du niveau des élèves les a contraints à différencier leurs enseignements à l'intérieur d'une même classe, ne serait-ce qu'en proposant une gradation des exercices proposés.

Ce problème est particulièrement crucial pour les savoirs cumulatifs, comme les mathématiques ou les langues vivantes pour lesquels les lacunes de certains élèves s'accumulent d'année en année, rendant impossible l'acquisition de nouvelles connaissances. Pour que les apprenants dans de telles situations puissent se remettre à flot, il faut donc revenir sur des notions qui ont déjà été abordées auparavant.

Notre objectif est donc de proposer à l'enseignant, mais aussi à l'apprenant, un système qui sélectionne un ensemble d'énoncés qui portent sur les notions que l'apprenant a à approfondir ou à découvrir. Cela présente deux avantages. D'une part, on peut ainsi espérer que l'apprenant devienne acteur de son processus d'apprentissage : s'il sait qu'il a des lacunes sur le théorème de Pythagore, il peut obtenir facilement une liste d'exercices abordant ce thème, ce qui lui permet de travailler de manière autonome. D'autre part, cela facilite le travail de l'enseignant qui peut sélectionner plus facilement les exercices qu'il veut faire résoudre à ses élèves. De plus, un tel système permet également une sélection automatique d'exercices, éventuellement en fonction du modèle de l'apprenant, permettant ainsi une planification dynamique des séances [FUT 00].

Pour ce faire, il faut que les exercices soient indexés par les propriétés utiles pour les résoudre. Cependant, si l'on veut proposer un nombre raisonnablement important d'exercices, il faut que cette indexation puisse se faire automatiquement. D'où notre idée d'utiliser un résolveur automatique de problèmes du domaine enseigné.

Disposant au Centre de Recherche en Informatique de Paris 5 d'un résolveur d'exercices de géométrie très performant, Argos [SPA 01] qui a résolu plus de 300 exercices de classe de 3$^{ème}$ jusqu'à la classe de terminale, nous avons appliqué les idées présentées ci-dessus au cas de la géométrie.

Dans le paragraphe suivant, nous présenterons succinctement Argos, puis nous introduirons les problèmes posés par l'indexation, en particulier nous expliciterons les choix que nous avons faits pour construire une ontologie du domaine adaptée à nos besoins. Ensuite nous présenterons le système réalisé et nous conclurons par quelques perspectives ouvertes par ce travail.

## 2 Argos

*Argos* a été développé par J-P. Spagnol à partir du démonstrateur *Muscadet*, mis au point par D. Pastre, qui au contraire de la plupart des autres démonstrateurs

automatiques, n'est pas basé sur le principe de résolution, mais sur des méthodes de déduction naturelle, qui tentent d'être plus proches du raisonnement humain, du cheminement cognitif du chercheur en phase de résolution de problème [PAS 02].

Les énoncés sont représentés à l'aide de prédicats[1], mais sans passer par une traduction de ces derniers en conjonction de clauses. De même, la trace de la preuve produite est lisible et compréhensible par un être humain, car elle se rapproche d'une démonstration « à la main », alors qu'une réfutation a peu de signification pour un non-spécialiste.

Argos est un système volumineux dont le comportement n'est pas toujours aisé à comprendre. A titre de comparaison, *Mentoniezh* comporte 51 règles [PY 96] écrites par le concepteur, alors qu'*Argos* en compte plusieurs milliers, générées automatiquement par le système à partir de connaissances données sous forme déclarative. Il peut s'attaquer à une large plage de problèmes : il est capable de résoudre des exercices allant du niveau de la classe de quatrième jusqu'à celui de terminale S, et en rédige une démonstration.

### *2.1 Nature des propriétés déclarées à Argos*

Les connaissances mathématiques sont exprimées déclarativement au système, sous forme d'implications. Par exemple, le théorème de Pythagore est donné sous la forme suivante :

```
propriete([general],[obligatoire],pythagore,
trRect(A,B,C) => longueur(B,C) * longueur(B,C) egal
longueur(A,C)*longueur(A,C)+longueur(B,A)*longueur(B,A
)
[On a: BC^2=AC^2+BA^2,car le triangle ABC est
rectangle en A]).
```

On reconnaît le théorème de Pythagore exprimé sous forme d'une implication, puis entre crochet, le modèle explicatif associé.

Puisque les connaissances sont exprimées sous forme d'implications, les théorèmes dont l'énoncé comporte une équivalence se voient scindés en deux : une propriété pour chaque sens de l'équivalence. D'autre part, les propriétés ne sont pas toujours données sous une forme déclarative. Ainsi la propriété `coordonneesPointApartirRelationVectorielle` permet, comme son nom l'indique, de calculer les coordonnées d'un point à partir d'une relation vectorielle. Mathématiquement, elle correspond à l'utilisation de l'égalité vectorielle $A\hat{M} = x\hat{V_1} + y\hat{V_2}$ pour calculer les coordonnées de *M*. Il ne s'agit pas d'un théorème mais de l'utilisation de différentes propriétés et définitions : relation de Chasles, coordonnées d'une combinaison linéaire de vecteurs et définition des

[1] Argos a été développé en SWI prolog

coordonnées. C'est typiquement une connaissance procédurale. L'introduction de telles propriétés dans le système *Argos* résulte certainement de son incapacité à résoudre certaines questions à partir des connaissances déclaratives seules. Nous verrons au paragraphe 3.1 en quoi la présence de savoirs opérationnalisés nous a posé quelques difficultés et comment nous les avons résolues.

### *2.2 Opérationnalisation des connaissances et recherche d'une preuve*

*Argos* opérationnalise ensuite ces connaissances en un certain nombre de règles. Par exemple pour le théorème de Pythagore, on obtient : `pythagoreDeduction` et `pythagoreDeductionCreation2`, que nous ne détaillons pas ici, la première conduisant au calcul de la « longueur inconnue » et la seconde créant les longueurs des côtés d'un triangle rectangle qui n'existent pas déjà pour le système, afin d'en permettre le calcul. Par la suite, le système ne fait plus référence aux propriétés introduites par l'utilisateur, il se sert uniquement des règles qu'il a construites. Ainsi le modèle explicatif associé à chaque propriété sert à générer des phrases type de rédaction qui apparaissent comme des paramètres des différentes règles créées.

La recherche de la preuve se déroule d'une manière analogue à ce qui se passe dans *Muscadet* : ce sont les méthodes de la déduction naturelle au sens de Bledsoe. La base de fait est divisée entre hypothèses et conclusions. Démontrer un fait revient à le faire passer de la catégorie de conclusion à celle d'hypothèse. Par exemple si le système doit traiter une implication du type, $p \rightarrow (A \wedge B)$, il cherchera à montrer $(p \rightarrow A) \wedge (p \rightarrow B)$, ou pour reprendre le cas du théorème de Pythagore, la règle `pythagoreDeduction` dit que si l'obtention de la longueur AC fait partie de la conclusion, que dans les hypothèses on a le fait ABC est un triangle rectangle en A, B ou C et les valeurs de AB et BC, alors il faut ajouter dans les hypothèses la valeur exacte de AC obtenue à l'aide du théorème de Pythagore.

Une fois que l'exercice est résolu, autrement dit que toutes les conclusions sont devenues des hypothèses dans la base de faits, *Argos* construit le graphe de la preuve à partir du dernier fait déduit en en cherchant les faits parents et ainsi de suite. Puis, une fois le graphe obtenu, la preuve est rédigée à l'aide des modèles explicatifs associés à chaque règle.

## 3 Indexer les exercices par les savoirs et savoir-faire utiles à leurs résolutions

### *3.1 Une ontologie du domaine*

Compte tenu de notre objectif, la question de l'organisation des connaissances en géométrie se pose indubitablement. La nécessité de disposer d'ontologies dans le domaine de l'enseignement assisté par ordinateur est largement avérée, tant pour la génération de matériel pédagogique à partir de documents de référence [DES 00]

que pour faciliter l'interopérabilité et la consultation des modules d'une plate-forme pédagogique. D'autre part, l'existence d'ontologie de domaine, par exemple la géométrie au niveau de l'enseignement secondaire, permettrait *« [une] construction incrémentale d'un environnement qui cumulerait, au fur et à mesure des fonctionnalités développées ailleurs »* ce qui est loin d'être le cas actuellement étant donnée la diversité des représentations des connaissances utilisées [GRA 96].

Pour clarifier notre propos, nous utiliserons les termes *règle*, *propriété* et *théorème* avec les sens suivants : les *théorèmes* sont ceux des mathématiques, au sens de la théorie de la démonstration, les *propriétés* sont les connaissances déclaratives données à *Argos* et les *règles* sont celles qu'il utilise pour résoudre le problème qui lui est posé.

### 3.1.1 Ontologie formelle

Les recherches en ontologie formelle, c'est-à-dire sur la nature des ontologies et leurs structures, se développent principalement dans deux directions.

1. B. Bachimont se base sur une théorie linguistique pour définir les ontologies. Sa méthode utilise comme concepts primitifs des termes obtenus par extraction terminologique à partir d'un corpus de textes jugés représentatifs du domaine et des relations qui les lient, puis il clarifie la signification de ces liens [BAC 01].

2. N. Guarino et C. Welty ont, quant à eux, défini des méta-propriétés pour les concepts afin de clarifier les liens de subsomption [GUA 02]. Ils théorisent ce qu'est l'essence d'un concept : une propriété est **essentielle** si elle est vérifiée dans tous les mondes possibles et **rigide** si elle est essentielle pour toutes ses instances. D'autre part, afin de pouvoir clairement différencier deux instances d'un même concept il faut définir clairement pour chacun d'entre eux un **critère d'identité**. Enfin la méta-propriété **unité** tend à opérer une distinction entre les instances qui constituent un tout et celles pour lesquelles ce n'est pas le cas.

Ces deux approches proposent tout autant des esquisses de méthodologies pour la construction des ontologies que des critères pour les valider.

### 3.1.2 Représentation des connaissances utilisées

Pour notre part, nous n'avons pas en fait construit d'ontologie de la géométrie. En effet, le domaine que nous devons représenter n'est pas la géométrie en elle-même, mais les exercices de géométrie. Et comme nous réalisons une indexation de ceux-ci à l'aide des savoirs mathématiques qui servent à les résoudre, ce sont eux qui sont les concepts primitifs de la représentation des connaissances que nous utilisons.

Cette question n'est pas anodine : on pourrait réaliser une ontologie des exercices de géométrie à l'aide des concepts qui interviennent dans leurs énoncés. En fait, c'est plus exactement l'ontologie des preuves en géométrie qui nous intéresse. De ce fait nous avons exclu les propriétés de calcul de notre indexation. Il nous faut donc nous pencher sur les théorèmes de géométrie et les liens qui existent entre eux.

Le principe d'identité développé par N. Guarino et C. Welty nous permet de clarifier ces relations. Nous considérons que les théorèmes de géométrie sont ceux reconnus comme tels dans l'enseignement secondaire. Si un théorème n'admet pas de cas particulier (considéré lui aussi comme un théorème), alors ce n'est pas un sous-concept de *Théorème*, mais une instance. Par exemple, si *Théorème de Pythagore* est un concept, alors le principe d'identité de *Théorème* doit permettre de distinguer deux instances de *Théorème de Pythagore*. L'identité de *Théorème* est celle des hypothèses et des conclusions : on distingue le *Théorème de Pythagore* du *Théorème de Thalès* parce que leurs hypothèses et leurs conclusions sont différentes. Or, pour distinguer deux instances de *Théorème de Pythagore*, il faut pouvoir en plus distinguer les jeux d'instanciation de leurs hypothèses et conclusions.

Le cas des théorèmes admettent des cas particuliers ayant aussi le statut de théorème, comme par exemple le théorème dit « de la droite des milieux » et sa réciproque, étudiés en classe de quatrième qui sont des cas particuliers du théorème de Thalès (et de sa réciproque) abordés en troisième, est plus délicat, et jusqu'à présent, notre indexation se fait uniquement en considérant le plus général des théorèmes. De plus, on pourrait introduire des concepts entre *Théorème* et ses instances[2], mais nous ne l'avons pas non plus encore fait.

D'autre part, nous utilisons un démonstrateur automatique pour effectuer cette indexation, nous devons donc articuler notre ontologie avec les propriétés déclarées dans *Argos*, qui ne sont pas exactement les théorèmes de la géométrie. Comme nous l'avons expliqué, les connaissances déclarées au système sont parfois en partie opérationnalisées. Pour reprendre l'exemple vu dans le chapitre précédent, la propriété : `coordonneesPointApartirRelationVectorielle` devrait tomber à la fois sous les concepts *Relation de Chasles* et *Coordonnées d'un vecteur*.

Dans le même esprit, certains théorèmes mathématiques sont découpés en plusieurs propriétés *Argos*. C'est en particulier le cas des théorèmes énoncés, dans l'enseignement secondaire, sous forme d'une équivalence, par exemple la caractérisation vectorielle du milieu se trouve « découpée » en deux propriétés, vectMilieu et reciproqueVectMilieu :

```
propriete([calculVectoriel],[obligatoire,seconde],vectMi
lieu,
milieu(A,B):I => vect(A,I) egalVect vect(I,B),
[I est le milieu de [A,B] donc,vect(A,I)= vect(I,B)]).
```

[2] Par exemple *Théorème de géométrie euclidienne* ou *Théorème de géométrie affine*, avec les difficultés que cela représente. En effet, au niveau secondaire, le théorème de Thalès (vectoriel ou non) fait intervenir la notion de longueur, directement ou non. Il devrait donc être catégorisé comme *Théorème de géométrie euclidienne*, or si l'on prend en compte la théorie des espaces vectoriels et celle des espaces affines qui en découle, le théorème de Thalès est un *Théorème de géométrie affine* : on l'énonce et le démontre sans utiliser de produit scalaire.

```
propriete([calculVectoriel],[obligatoire,seconde],recipr
oque VectMilieu,
vect(A,I) egalVect vect(I,B) =>     milieu(A,B):I,
[I est le milieu de [A,B] car vect(A,I)= vect(I,B)]).
```

Par conséquent, l'utilisation des propriétés d'*Argos* pour réaliser une indexation d'exercices de géométrie n'est pas tout à fait pertinente. Un élève ayant à travailler sur la caractérisation vectorielle du milieu d'un segment ne doit pas avoir à faire le choix entre des exercices utilisant `vectMilieu` ou `reciproqueVectMilieu`. Nous catégorisons donc les propriétés d'*Argos* à l'aide d'une ontologie, assez sommaire, des théorèmes de géométrie au niveau de l'enseignement secondaire, c'est à dire que nous avons utilisé une les théorèmes de géométrie pour réaliser l'ontologie des propriétés d'*Argos*. Nous avons ainsi catégorisé environ 150 propriétés à l'aide de 75 théorèmes.

Cependant, nous avons vu que *théorème de Pythagore* est une instance, un objet dans l'ontologie des théorèmes de la géométrie, or pour catégoriser les propriétés d*'Argos* nous le considérons comme un concept. Nous pensons que d'un point de vue théorique, nous avons affaire à un changement de degré de généralité des ontologies considérées. Les concepts de l'ontologie des propriétés d'*Argos* peuvent être des objets de l'ontologie des théorèmes de géométrie, de même que le concept *Humain* de l'ontologie des êtres vivant dans un appartement est un objet de celle des espèces.

Pour résumer, la représentation des connaissances que nous utilisons est la suivante. Un exercice[3] est représenté par ses attributs : son nom et les différents théorèmes de géométrie du secondaire qui servent à sa résolution.

### *3.2 Modifications apportées à Argos et obtention du fichier d'indexation*

#### 3.2.1 Le problème de la saturation

Initialement, nous comptions utiliser *Argos* pour obtenir plusieurs solutions différentes d'un même exercice, ce qui aurait permis une indexation plus large. Nous allons exposer dans ce paragraphe les difficultés que nous avons rencontrées et qui nous ont conduit à abandonner ce projet.

Nous avons réfléchi aux modifications envisageables pour faire saturer le démonstrateur, sans modifier les connaissances du système, car des expériences dans ce sens s'étaient montrées infructueuses. Nous avons pensé à empêcher le déclenchement avec le même jeu d'instanciation d'une règle ayant déjà été utilisée. Cependant, deux preuves différentes peuvent tout à fait nécessiter l'utilisation d'une même règle avec le même jeu d'instanciation, surtout compte tenu du fait que les règles sont l'expression de propriétés aussi simples et essentielles que la transitivité de l'égalité.

---

[3] Dans le sens « un énoncé avec une de ses solutions ».

Finalement, lors d'une discussion, Dominique Pastre, conceptrice de *Muscadet*, dont *Argos* est issu, nous a expliqué que les démonstrateurs basés sur la déduction naturelle ne sont pas fait pour être saturés, c'est même contradictoire avec leur conception qui vise à éviter une systématique aveugle.

### 3.2.2 Création du lien entre règles et propriétés

Comme nous l'avons expliqué auparavant, *Argos*, à partir des connaissances qui lui sont fournies sous forme d'implications, crée dynamiquement des règles qui interviennent ensuite lors de la résolution des problèmes qui lui sont posés. Les noms de ces règles sont aussi générés automatiquement, et lors de la construction de la preuve et de sa rédaction, ce sont eux qui sont utilisés, il n'est plus fait référence aux propriétés.

Parmi les éléments mémorisés par *Argos* lors de la résolution d'un exercice, il y a la liste des règles qui ont été utiles, sous la forme du prédicat `listeReglesUtiles(nomExercice,[regle1,…,regleN])`.

Le nom des règles est obtenu à partir de celui de la propriété dont elle découle, nom auquel le système ajoute un suffixe, par exemple `Deduction`, nous pouvions donc récupérer la liste des propriétés utilisées en analysant les chaînes `regle1` à `regleN`. Cependant, cette analyse devrait s'effectuer à chaque indexation d'un nouvel énoncé, il est donc préférable de mémoriser le lien entre une règle et la propriété dont elle découle. Ici encore, cette mémorisation aurait pu se faire a posteriori, une fois l'ensemble des règles généré. Toutefois il nous a semblé plus pratique et plus rapide d'opérer directement lors de cette phase de génération.

Dans un premier temps, nous avons tenté de faire cette mémorisation au moment de la création de la règle, mais les résultats obtenus n'étaient pas satisfaisants. La traçabilité n'étant par particulièrement aisée en *Prolog*, nous avons décidé, après plusieurs modifications infructueuses, de changer de stratégie et d'effectuer la mémorisation du lien entre une règle et la propriété dont elle découle au moment de la création du nom de cette règle.

Pour ce faire, on ajoute le but `LienProprieteRegle(P,R)`, comme sous-but de la création du nom d'une règle, `créer_nom_regle`. Ainsi dès qu'une nouvelle règle `R` est créée et nommée à partir d'une propriété `P`, le fait `associe_propriete_regle(P,R)` est écrit dans le fichier `lienProprietesRegles`. Ce dernier contient donc les liens entre propriétés et règles, sous forme de fait *Prolog*.

### 3.2.3 Lien entre propriétés et théorèmes

Les liens entre les propriétés *Argos* et les théorèmes sont mémorisés sous forme de faits *Prolog*.

Par exemple, `lienProprieteTheoreme(vectMilieu,vectMilieu)` et `lienProprieteTheoreme(reciproqueVectMilieu,vectMilieu)` expriment le lien d'instanciation entre les propriétés vectMilieu et

reciproqueVectMilieu et le théorème de caractérisation vectorielle du milieu d'un segment[4].

Ces liens permettent par conséquent de relier les propriétés déclarées au système aux théorèmes susceptibles d'être utilisés par les élèves.

### 3.2.4 Obtention de la liste des théorèmes utiles

Exercice par exercice, on obtient la liste des théorèmes qui ont servi à sa démonstration de la manière suivante : pour toute règle apparaissant dans la liste des règles utiles, le système cherche la propriété à laquelle elle est associée, puis le théorème duquel dépend cette propriété. On l'ajoute alors à la liste des théorèmes utiles, mémorisée à l'aide du prédicat `listeProprietesUtiles(Exercice,Liste)`. Il ne reste alors plus qu'à formater l'écriture pour obtenir des liens indexables par la base de données.

Résumons le processus : à partir de la résolution d'un exercice nous récupérons les règles utilisées par *Argos*, puis nous obtenons les propriétés du système dont elles découlent, enfin à l'aide de notre ontologie nous en tirons les théorèmes mathématiques ayant servi pour cette résolution, qui permettent l'indexation dans une base de données.

### 3.2.5 Une heuristique sur la difficulté des énoncés proposés (non implémentée)

Il serait intéressant de raffiner l'indexation que nous obtenons en tenant compte de la difficulté des énoncés proposés. L'évaluation de celle-ci reposant évidemment sur la preuve générée par *Argos*.

Tout d'abord à partir du moment où l'on dispose d'une ontologie des théorèmes de géométrie dans l'enseignement secondaire, on peut définir pour chaque exercice la notion de **niveau obligatoire.** A chaque théorème, on peut associer la classe dans laquelle il est enseigné pour la première fois, on obtient alors pour chaque exercice une liste de classes qui dépend des théorèmes servant à sa résolution, le niveau le plus élevé de ses classes étant alors le **niveau obligatoire** de l'exercice.

Cependant, cette première information sur la difficulté d'un énoncé est assez grossière. Il faudrait la compléter d'une heuristique prenant en compte le nombre de pas de la démonstration proposée, la complexité des théorèmes utilisés et l'éventuelle nécessité de créer des objets. Par exemple :

$$D(e) = np(e) + \sum_{th_i \in Th(e)} d(th_i) + 5noc(e)$$

[4] Il est peu judicieux de choisir un même nom pour un concept et une de ses instances. Il faudrait donc renommer les concepts, afin d'éviter toute confusion.

où $D(e)$ est la difficulté de l'exercice $e$, $Th(e)$ l'ensemble des théorèmes nécessaires à sa résolution, $np(e)$ le nombre de pas de sa preuve, $noc(e)$ le nombre d'objets crées lors de celle-ci et $d(th_i)$ la difficulté du théorème $th_i$.

Le choix proposé à l'élève ne pouvant se résumer à l'entrée d'une valeur pour la difficulté, il conviendrait alors de définir des niveaux de difficulté selon le niveau obligatoire et $D(e)$, par exemple : facile, peu difficile, difficile et très difficile.

## 4 Présentation du système

Dans la mesure où *Argos* a été rédigé en *Prolog*, les modifications et les ajouts que nous lui apportons se font également dans ce langage. *Prolog* est particulièrement efficace pour rechercher les jeux d'instanciations d'un prédicat, cependant, des IHM graphiques aisément utilisables par des novices ne sont pas facilement disponibles. Il nous paraît contradictoire de vouloir créer un système utilisable par des utilisateurs néophytes et de leur faire rédiger des requêtes en *Prolog*.

De ce fait, nous avons choisi d'indexer les exercices dans une base de données, puis d'interroger celle-ci par l'intermédiaire d'une interface graphique. Nous nous sommes orientés vers la solution *php/mysql*, solution devenue classique pour construire des sites web dynamiques interrogeant une base de données.

Venons-en maintenant au fonctionnement général du système. A partir des informations obtenues par *Argos* lors de la résolution d'un exercice, le système génère un fichier nommé `listeProprietesUtiles` exploitable par la base de données. La consultation de celle-ci se fait à l'aide de la **feuille d'exercice**, développée en *php*, qui permet à l'élève ou au professeur de consulter la base de données de manière simple, sans connaissance sur les langages de requête.

L'utilisateur se trouve face à un menu lui permettant de choisir le théorème sur lequel il désire travailler. Son choix est traduit sous forme de requête *mysql*. Le résultat de cette requête, les noms des exercices et leurs énoncés sont affichés sur la page suivante.

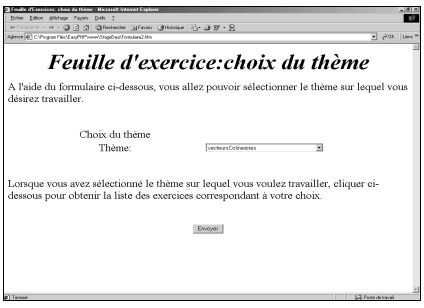


Figure 1: **Menu de sélection du théorème**

## 5 Bilan et perspectives

Ce projet a pour objectif de générer une feuille d'exercices à l'aide d'une indexation automatique de ces derniers compte-tenu des théorèmes utilisés pour leurs démonstrations.

Nous n'avons pu réaliser l'indexation que sur une seule preuve par énoncé, étant donnée la difficulté à faire trouver à *Argos* plusieurs démonstrations pour un même exercice.

Pour permettre une réelle adaptation à l'apprenant, il faudrait permettre une indexation en fonction du niveau et de la difficulté de la preuve, celle-ci pouvant être mesurée à l'aide d'une heuristique prenant en compte le nombre de pas de la démonstration, le nombre d'objets créés et la complexité des théorèmes utilisés, comme nous l'avons expliqué.

Par ailleurs, l'ontologie des théorèmes de géométrie pourrait être ramifiée en adaptant sa granularité selon les différents niveaux scolaires : plus précisément, une ontologie des théorèmes de géométrie de l'enseignement secondaire comportant suffisamment de concepts doit permettre de modéliser les connaissances avec la granularité nécessaire. Par exemple, le théorème de Pythagore direct et sa réciproque seraient des grains pour l'indexation au niveau troisième, le concept parent de théorème de Pythagore (construit extensionnellement à l'aide des deux précédents) servant pour celle de première S et terminale S, tout en étant inclus dans une même ontologie.

Enfin, plusieurs noms de concepts de l'ontologie devraient être modifiés afin d'éviter toute confusion.

La réalisation de ce système montre la faisabilité de ce type de projet, qui devrait naturellement émerger lors de la conception d'EIAH visant l'apprentissage de méthode de résolution de problèmes. En effet, pour permettre à l'apprenant de travailler efficacement sur un nombre important de problèmes, il faut qu'ils soient indexés, et du fait de leur nombre cette indexation doit pouvoir se faire automatiquement.

A partir du moment où un EIAH intègre un résolveur de problèmes du domaine, on peut envisager de lui ajouter un module d'indexation et de génération de feuille d'exercices tel que celui que nous avons développé. Mais pour ce faire, c'est-à-dire pour permettre une réelle interopérabilité des systèmes développés, il est nécessaire de construire et de diffuser des ontologies, problématique abordée par de nombreux auteurs (voir par exemple R.Mizoguchi et J.Bourdeau [MIZ 00] ou C. Desmoulins et M. Grandbastien [DES 00] ).

**Remerciements**

Les auteurs tiennent à remercier Dominique Pastre pour ses conseils sur le fonctionnement de *Muscadet*.

## Bibliographie

[BAC 01] BACHIMONT B., « Modélisation linguistique et modélisation logique des ontologies: l'apport de l'ontologie formelle », *Ingénierie des Connaissances*, 2001.

[DES 00] DESMOULINS C. et GRANDBASTIEN M., « Indexer des documents techniques à l'aide d'ontologies pour construire des manuels de formation professionnelle », *Actes, IC2000*, 2000.

[FUT 00] FUTTERSACK M., LABAT J-M., « Quelle planification pédagogique dans les EIAH ? », *Sciences et Techniques Educatives*, vol. 7-n°1/2000.

[GRA 96] GRANDBASTIEN M., « Introduction », *Sciences et Techniques Educatives,* vol. 3-n°2/1996, p.145-155.

[GUA 02] GUARINO N. et WELTY C., « Evaluating Ontological Decision with ONTOCLEAN », *communications of the ACM, February 2002/Vol.45.No2.*

[MIZ 00] MIZOGUCHI R. et BOURDEAU J., « Using Ontological Engineering to Overcome Common AI-ED Problems », *IJAIE, vol. 11 n°2*, 2000.

[PAS 02] PASTRE D., « Strong and weak points of the MUSCADET theorem prover-exemples from CASC-JC », *AI Communications* 15, IOS Press 2002.

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

[SOL 91] SOLOWAY E., « How the Nintendo Generation Learns », *Communications of the ACM*, vol. 34 n°9, 1991.

[SPA 01] SPAGNOL J-P., Automatisation du raisonnement et de la rédaction de preuves en géométrie de l'enseignement secondaire, thèse de doctorat, Université René Descartes - Paris V, 2001.

## Références sur le Web :

Sur les ontologies :

www.ladseb.pd.cnr.it/infor/Ontology/ontology.html, 2002.

www.eisanken.osaka-u.ac.jp/english, 2003.

Sur le langage *SWI-Polog* :

www.ufr-info-p7.jussieu.fr/prolog, 2002.

Sur *Argos* et *Muscadet* :

http://perso.club-internet.fr/jps0123/, 2002.

www.math-info.univ-paris5.fr/~pastre/muscadet/muscadet.html, 2002.