



HAL
open science

Automatically generated exercise pages

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

► **To cite this version:**

Mathieu Hibou, Jean-Marc Labat, Jean-Pierre Spagnol. Automatically generated exercise pages. AIED 2003 conference, 2003, Sydney, Australia. 7 p. hal-00190311

HAL Id: hal-00190311

<https://telearn.hal.science/hal-00190311>

Submitted on 23 Nov 2007

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.

Automatically generated exercise pages.

Mathieu HIBOU, Jean-Marc LABAT, Jean-Pierre SPAGNOL
Université Paris V René Descartes, CRIP5, SAFE-SBC.
45 rue des Saints-Pères 75270 Paris Cedex 06
{*Mathieu.Hibou, Jean-Marc.Labat*}@math-info.univ-paris5.fr

Abstract : In order to generate an exercise page on a particular knowledge, exercises have to be indexed according to the knowledge useful to solve them. Hence, the main idea of this project is to use a problem solver, Argos, to help us deal with this indexation.

Argos aims not only at solving geometry problems, but also writes their proofs just as a teacher or a pupil would. Consequently, the meaningful data have to be extracted from the proofs found by the solver.

Argos automatically generates rules from the properties given to it in a declarative way. We have linked the rules with geometric properties in order to deal with them, because the granularity of Argos' rules is too fine.

The useful theorems are stored in a database, so it is easy to build an exercise page on a particular subject (e.g. Pythagoras' theorem) by interrogating it.

Hence the system we propose provides an assistance to the teaching of the curriculum or to the planning of activities in an Intelligent Tutoring System.

keywords: Ontology, Metadata, Automatic demonstration, Knowledge Representation, Indexation.

Introduction

There has been a lot of changes in the profile of high-school pupil over the last years: not only as regard as their social and cultural origins, but also as to their learning and working methods. To cope with these evolutions, teachers have had to change their pedagogical methods. The lack of homogeneity of the classes has compelled them to diversify their teaching within a single group, even if it just consists in giving level-adapted exercises on the same topic.

This problem is particularly crucial to cumulative knowledge, as mathematics or foreign languages: year after year some pupils accumulate deficiencies that make any new acquisition impossible. It is not sufficient to give simpler exercises or to add intermediary questions to enable a learner to progress when the block lies upstream. In order to help a learner deal with such a situation, it is necessary to go back to notions taught a long time before.

Thus, our purpose is to offer teachers and learners a system which could select a array of exercises focusing on a particular topic that needs to be probed or discovered. This approach presents two advantages. On the one hand, learners would become actors of their learning process, and their relationship with the computer is important, as mentioned by Soloway in [1]. Knowing he/she has difficulties with Thales' theorem, a pupil would be able to have access easily to a list of exercises using it, and he/she could thus work autonomously. On the other hand, it facilitates teachers' work by giving them the possibility to select categorised exercises. Moreover, such a system could be of great help in a dynamic scheduling of sessions in an ITS, possibly according to the learner's model [2].

Hence, exercises need to be index-linked by the properties that help to solve them. However, in order to propose a reasonable amount of exercises, this indexation has to be done automatically. This explains why we decided to use an automatic solver of the field taught.

Since we have at our disposal at the Centre de Recherche en Informatique de Paris 5 (research centre in computer science) in the university René Descartes a high performance problem solver in the field of geometry, Argos [3], which has solved more than 300 exercises up to the level of the French “baccalauréat” (approximately from the 8th to the 12th form in the British educational system), we have applied the above mentioned idea to geometry.

In this paper we first present Argos briefly, and we introduce the problems raised by indexing. Finally we present the system we have developed, and we conclude by giving some perspectives offered by this work.

1. Argos

Argos was developed by J-P.Spagnol. It is derived from Muscadet, a problem solver conceived by D.Pastre [4]. Contrary to most of the other problem solvers, it is not based on the resolution principle, but on methods of natural deduction, trying to be closer to the human reasoning, that is to say closer to researchers’ cognitive progression.

Predicates are used to represent the problems, but there is no need to translate them into conjunctions of clauses. In the same way the traces of the proofs produced by the system are readable and understandable; they are close to a demonstration made by a human, whereas a refutation has no meaning for a non-specialist. Therefore, Argos is well adapted to learning environments [5].

Argos is a voluminous system and its behaviour is therefore not always easy to understand. For instance, Mentoniez deals with 51 rules [6] written by its developers, whereas Argos counts several thousand rules, automatically generated by the system from the knowledge given in a declarative way. Moreover, this system is able to solve rather difficult problems and to write their demonstrations.

1.1. Properties declared to Argos.

The mathematical knowledge is expressed declaratively to the system, as logical implications. For instance, Pythagoras’ theorem is given as follows:

```
propriete([general],pythagore,
RightTriangle(A,B,C) => length(B,C) * length(B,C) equal
length(A,C)*length(A,C)+length(B,A)*length(B,A)
[We have: BC^2=AC^2+BA^2,because ABC is a right-angled
triangle in A]).
```

We can easily recognise Pythagoras’ theorem expressed as an implication, then , in brackets, the explanatory model, that is to say the sentence that will be used when the demonstration of the proof is written by Argos.

Because of the way knowledge is expressed in the system, theorems containing equivalences have to be divided in two parts: one property for each implication of the equivalence. Moreover, the properties are not always entirely given in a declarative way. Thus, the property `pointCoordinatesUsingVectorRelation` enables to get the coordinates of a point using a vector relation. Mathematically, it consists in using the equality $\overrightarrow{AM} = \alpha\vec{U} + \beta\vec{V}$ to get the coordinates of the point M , knowing A , α , β and the vectors \vec{U} and \vec{V} . This is not exactly a theorem. It is a combination of definitions and properties: Chasles’ relation, the coordinates of a linear combination of vectors, and of course the definition of the coordinates. Typically, this is a practical skill, a know-how that should be mastered by the pupils. Know-hows are methods, skills: they are operating knowledge. Similarly `thalesEquation` describes how to get an unknown length using the equality obtained with

Thales' theorem. The introduction of such properties in Argos is due to the impossibility for the system to solve some questions only by using declarative knowledge, so, in a way, it has the same difficulties as a human learner. In paragraph 3.1 we explain how we try to deal with the presence of operating knowledge in Argos.

1.2. Knowledge operationalisation and proof searching.

With each property, Argos creates rules. For instance, when dealing with Pythagoras' theorem, the system generates the rules `pythagorasDeduction` and `pythagorasDeductionCreation2`, which we do not explicit. The first leads to the calculation of the "unknown length", while the second creates the lengths of the right-angled triangle sides that do not exist already, in order to enable their computation. Afterwards, the system does not mention the properties given by the expert anymore, it uses only the generated rules to perform its search of a proof.

The search of a proof progresses just as in *Muscadet* : the methods of Bledsoe's natural deduction are used as follows. The facts base is divided in two parts: hypotheses and conclusions. Proving a fact consists in leading it from the category of conclusion to the category of hypothesis. The system proceeds as similarly to human beings as possible. For instance, looking again at the case of the well-known Pythagoras' theorem, the rule `pythagoreDeduction` says that if the calculation of the length AC is one of the conclusions, if the fact that ABC has a right angle in A, B or C and the length AB and BC are present in the hypothesis, then, the exact value of BC, obtained with Pythagoras' theorem, must be added to the hypothesis.

Once the exercise is solved, that is to say, once all the conclusions have become hypotheses, Argos builds the graph of the proof: beginning with the last deduced fact, it looks for its parents and so on. Then, once the graph is obtained, the proof is written using the typical sentences that are attached to each rule, drawn from those declared with the properties.

2. Indexing exercises using the knowledge useful to solve them.

2.1 Knowledge representation

We aim at getting an automatic indexation of exercises in the field of geometry , so we have to lay the stress on knowledge representation in geometry. The need of ontologies in the field of ITS is widely recognised. Not only to generate pedagogical resources from documents [7], but also to facilitate interoperability and data management in pedagogical platforms. Moreover, the question of the re-use is central for such topics, in that the existence of domain ontologies should enable an incremental conception of a system cumulating more and more functionalities [8]. This is not the case yet because of the diversity of knowledge representations used.

In order to clarify our words, we will use the terms "rule", "property", and "theorem" in the following senses: the theorems are mathematical statements, the properties are the statements declared to Argos and the rules used to solve the exercises are automatically generated by the system.

In fact we have not build an ontology of geometry. The domain we have to represent is not geometry in itself, but the geometry exercises. As we have to proceed to an indexation using mathematical theorems, these are the primitive concepts of the knowledge representation that we use.

This is not a trivial topic: one might have built an ontology of geometry exercises using the concepts occurring in their descriptions. Actually, we are interested in indexing the geometric proofs. That's the reason why the calculation properties are not relevant to our indexation.

Moreover, as we are using an automatic solver to proceed to this indexation, we have to consider the knowledge representation of theorems already existing, at least implicitly in Argos. Nevertheless, the properties declared in Argos are not exactly mathematical theorems. We have already explained that the knowledge given to the system is sometimes partly operationalised. For instance the property `pointCoordinatesUsingVectorRelation` should be simultaneously subsumed by the concepts Chasles' Relation and Vector Coordinates.

In the same way, some theorems are split in Argos in different properties, as theorems expressed by a logical equivalence. For instance, the vectorial characterisation of the midpoint is split in two properties, `vectMidpoint` and `reciprocalVectMidPoint`:

```
property([vectorialCalculus],vectMidpoint,
midpoint(A,B):I => vect(A,I) equalVect vect(I,B),
[I is the midpoint of segment [A,B] hence,vect(A,I)=
vect(I,B)]).
```

```
property([vectorialCalculus],reciprocalVectMidpoint,
vect(A,I) equalVect vect(I,B) => midpoint(A,B):I,
[I is the midpoint of segment [A,B] because vect(A,I)=
vect(I,B)]).
```

Consequently, using Argos properties to realise the indexation does not appear really judicious. A pupil that has to work on the vectorial characterisation of the midpoint should not have to choose between exercises using `VectMidpoint` and those using `reciprocalVectMidpoint`. Moreover, the granularity of the knowledge representation in Argos does not always fit the level of the pupils it is intended for. We categorise the 150 Argos properties using 75 concepts in a basic ontology of secondary education theorems.

To sum up, the knowledge representation that we use is the following: any exercise is represented by its slots: its name and the theorems that are useful to solve it. These theorems are embedded in an basic ontology of secondary education theorems.

2.2 Argos' modifications¹

We had to modify the Argos code in order to make it generate a file containing the predicates that were necessary to get the indexation file.

2.2.1 Links between rules and properties

Argos dynamically creates rules that will be used to solve problems. It creates the names of each rule according to the name of the property it comes from by adding a suffix or a prefix.

¹ Initially we thought about using Argos to get different solutions to the same problem, in order to make a wider indexation, but we had to give up: during an informal conversation, Dominique Pastre, who developed Muscadet, from which Argos is derived, explained to us that solvers based on Bledsoe's natural deduction are not meant to be saturated. It is contradictory with their conception.

When writing the proof, the system does not make reference to the properties, but only to the rules.

The list of the rules used to solve an exercise is memorised by Argos as a logical predicate: `listUsefulRules(nameExercise, [rule1, ..., ruleN])`.

By analysing the strings `rule1` to `ruleN` we should be able to get the list of useful properties for any given exercise. However, this analysis should be done for each indexation, and that is the reason why we would rather memorise the links between rules and properties directly each time a rule is generated.

For different technical reasons not detailed in this paper, it is more efficient, due to the Argos architecture, to memorise this link when the name of the rule is created. We add the goal `linkPropertyRule(P, R)` as a sub-goal of the creation of the rule's name. Hence as soon as a new rule `R` is created and named from a property `P`, the fact `associate_property_rule(P, R)` is written in the file `linksPropertiesRules`. Consequently, this file contains the links between properties and rules in Prolog fact form.

2.2.2 Links between properties and theorem

As the links between rules and properties, the links between properties and theorems are memorised in Prolog fact form. For instance, `linkPropertyTheorem(reciprocalVectMidPoint, vectMidPoint)` expresses the instantiation link between the property `reciprocalVectMidPoint` and the vectorial characterisation of the midpoint.

2.2.3 the list of useful theorems

We developed different procedures to get the list of useful theorems from the facts memorised. The system proceeds as follows: for any rule appearing in the list of useful rules, it searches to which property the rule is linked, then the theorem from which that property comes. This theorem is then added to the list of useful theorems, that will be formatted in order to be stored in a database.

3. The system

The modifications and additions we made were done in Prolog as it is the language in which Argos has been developed. For this language, easy to use GUI are not easily available, and because it seems contradictory to create a system for secondary school pupils and to make them write Prolog queries, we have decided to index the exercises in a database, then to interrogate it through a graphic interface.

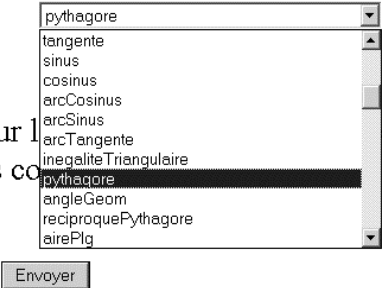
From the data obtained by Argos, the system creates a file storable in the database. Through a web page developed in php, the pupil or the teacher questions the database as simply as possible. The user has to choose a theorem in a menu list (fig.1), then his/her choice is translated into a mysql query.

A l'aide du formulaire ci-dessous, vous allez pouvoir sélectionner le thème sur lequel vous désirez travailler.

Choix du thème

Thème:

Lorsque vous avez sélectionné le thème sur la liste ci-dessous pour obtenir la liste des exercices correspondants, cliquer ci-dessous.



Envoyer

Figure 1: menu list.

The result of this query, problems names and descriptions are displayed on the next page (fig.2). The descriptions are given in the formal language that is used to describe the problems to *Argos*, which is rather easy to understand.

Exercice:declc28
Enoncé:
trRectIsocele(a,b,c) et valeurExacte(longueur(b,a),2*rac(2)) => calculer(longueur(c,b))

Exercice:th229
Enoncé:
m appCercle cercle(o,4) et projeteOrthog(m,droite(a,b)):h et longueur(o,h) egal 2.4 et diametre(a,b,droite(a,b),cercle) => calculer(longueur(m,h))

Figure 2: problems descriptions.

4. Results and prospects

The aim of this project is to generate an exercise page from an automatic indexation of geometry exercises, considering the theorems useful to solve them.

In order to enable a real adaptation to the learner, we should take the level and the difficulty of a proof into consideration for the indexation. This could be done by using a heuristic taking into account the number of steps in the demonstration, the number of objects created by the system and the complexity of the theorems used.

Otherwise the ontology of geometric theorems could be ramified adapting its granularity to different school levels.

We think that the topic of this paper is rather general. When developing an ITS to teach problem solving, the idea of building an index of exercises should naturally emerge, and in order to propose to the user a large number of problems, these exercises need to be automatically index-linked. Thus, any ITS including a problem solver could get an indexation add-on program and an exercise page generator similar to the one we have developed.

However to make it happen, that is to say to enable a real interoperability between systems, it is necessary to build and disseminate ontologies, a problematic treated by many authors, for example R.Mizoguchi and J.Bourdeau [9] or C.Desmoulins and M.Grandbastien [7].

Acknowledgements

The authors would like to thank Dominique Pastre for her advice about *Muscadet*.

References

- [1] SOLOWAY E., « How the Nintendo Generation Learns », *Communications of the ACM*, vol. 34 n°9, 1991.
- [2] FUTTERSACK M., LABAT J-M., « Quelle planification pédagogique dans les EIAH ? », *Sciences et Techniques Educatives*, vol. 7-n°1/2000.
- [3] 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.
- [4] PASTRE D., « Strong and weak points of the MUSCADET theorem prover-exemples from CASC-JC », *AI Communications* 15, IOS Press 2002.
- [5] SPAGNOL J-P ., « Modelisation and Automation of Reasoning in Geometry. The ARGOS System: a Learning Companion for High-School Pupils », *Proceedings of the 6th International Conference, ITS 2002*, Springer Verlag 2002.
- [6] PY D., « Aide à la démonstration en géométrie : le projet Mentoniez », *Sciences et Techniques Educatives*, vol. 3-n°2/1996.
- [7] DESMOULINS C. et GRANDBASTIEN M., « Indexer des documents techniques à l'aide d'ontologies pour construire des manuels de formation professionnelle », *Actes, IC2000*, 2000.
- [8] GRANDBASTIEN M., « Introduction », *Sciences et Techniques Educatives*, vol. 3-n°2/1996, p.145-155.
- [9] MIZOGUCHI R. et BOURDEAU J., « Using Ontological Engineering to Overcome Common AI-ED Problems », *IJAIED*, vol. 11 n°2, 2000.