



Repositorio Institucional de la Universidad Autónoma de Madrid

<https://repositorio.uam.es>

Esta es la **versión de autor** de la comunicación de congreso publicada en:
This is an **author produced version** of a paper published in:

Mathematical Knowledge Management: Third International Conference, MKM 2004, Białowieża, Poland, September 19-21, 2004. Proceedings. Lecture Notes in Computer Science, Volumen 3119. Springer 2004. 265-275

DOI: http://dx.doi.org/10.1007/978-3-540-27818-4_19

Copyright: © 2004 Springer-Verlag

El acceso a la versión del editor puede requerir la suscripción del recurso
Access to the published version may require subscription

Modeling Interactivity for Mathematics Learning by Demonstration

Miguel A. Mora, Roberto Moriyón, and Francisco Saiz

Escuela Politécnica Superior
Universidad Autónoma de Madrid
28049 Madrid, Spain
{Miguel.Mora, Roberto.Moriyon, Francisco.Saiz}@uam.es

Abstract. In this paper we present a mechanism for adding interactivity to static mathematical documents, which become interactive programs that allow students to practice the resolution of problems that involve symbolic computations. The designers that use this mechanism can work in the same environment used by students when solving the problems, and they don't need to know any programming language. The original problems can also be generalized, and sets of similar problems that can be solved using the same methods can be generated automatically. The mechanism described has been implemented in a computer system that has also collaborative capabilities.

1 Introduction

Explanations about Mathematics methods, or examples about applications of these methods, are usually included in static documents in a style similar to textbook pages. These documents lack an important feature that is desirable in a learning environment, namely interactivity, which allows learners to get involved in the learning process in a more participative way. In this paper we present a mechanism for adding interactivity to static documents, and at the same time structuring in a hierarchical manner the presentation of the information contained in them.

In order to achieve the desired interactivity, it is essential to have an adequate level of structuring of the documents to be used. Mathematical documents usually consist of pieces of information with different levels of structure, such as formulae, graphics or text. This information is not usually structured according to rigid patterns, as opposed to computer-generated pieces of information, since for example it might correspond to pages within Mathematics textbooks. In this context, semantic web efforts are well known to be addressed to the structuring of the information contained in documents, both in more general settings, [?], and in more particular ones, e.g. in Mathematics [?], where they usually rely on Mathematics representation languages, like MathML, [?].

Additionally, the parts that form mathematical documents, such as formulae or parts of them, are usually deeply interrelated. For instance, an explanation

about a resolution method usually starts with some initial formulae and continues with a series of steps using them to generate new ones, which in turn can be utilized for new steps. A classical approach in order to deal with this issue, even though it should be mentioned that it is applied to purely numeric contexts, is the spreadsheet mechanism. On the other hand, formula evaluation is another major issue. For instance, it is usual to find the same value in a document several times with different degrees of evaluation, e.g. an arithmetical expression with parenthesis like $(3+1)^2$ that is evaluated by first evaluating the expressions between them, giving $3^2 = 9$. This also happens in the case of evaluations carried out by Computer Algebra Systems (CAS), [?], [?], where some mathematical expressions are completely evaluated. Sometimes these evaluations correspond to the resolution of a subproblem attached to a given resolution method. Besides, the degree of evaluation of the formulae is an issue that is not under a user's control.

Static documents basically serve as textbooks, with the added functionality of searching based on indexing, [?]. However, students prefer to get involved by making actions and receiving feedback from the documents. A basic example would be the case of spreadsheet documents where they not only can browse the information contained in them but they can also change values in certain cells, thereby receiving feedback showing them the result of the update of the constraints.

A richer case of interactivity corresponds to the situation where students are able to enter expressions and answers to prompts from a computer system, or to choose options presented to them. For instance, at those decision points a student can be asked to choose a resolution method and accordingly the system will enforce it. This is the case of MathEdu, [?], which provides a rich interaction capacity built on Mathematica, [?], or Calculus Machina, [?], another advanced tutoring system with other features comparable to those of MathEdu. Still, there exists today a lack of authoring tools that allow teachers to establish this kind of interactivity in a simpler way, specially starting from static documents which can serve as sources for establishing the questions. Moreover, the interactive applications developed with these authoring tools should adapt to different situations by including conditions for the applicability of the operations involved in a resolution method.

In this paper we present a mechanism of the type described above for creating a model for the interaction with the student during the resolution of problems of Mathematics. This mechanism has been implemented in ConsMath, [?], a computer system that can be used while learning parts of Mathematics that involve symbolic computations. The major features of the system are the following ones:

- ConsMath integrates both an authoring tool and an execution tool. By using the WYSIWYG authoring tool, teachers design interactive tutoring applications in an intuitive and simple way, due to the similarity of the environments used by teachers and students.
- The design process carried out by teachers is done by using programming by demonstration methods, [?], which is an important technique belonging

to the field of Human-Computer-Interaction. Thus, at each instant designers work with specific examples of resolution methods, and they identify those examples as particular cases of more general methods generating subsequently a more general and abstract model.

- By using ConsMath, an interactive application for the resolution of a certain pattern problem can be built in a simple way starting from a static document that shows the resolution for a particular problem. This initial document can be generated by the designer either from scratch or by using the ConsMath editor, which is able to import documents that include mathematical formulae written in the MathML language.
- ConsMath has been built using a collaborative framework, [?]. As a consequence of this, students can learn collaboratively and teachers can interactively monitor their actions, based on the interaction model created by teachers.

We have tested how ConsMath can be used for the design of interactive sets of problems. These tests have been performed by two math teachers. A collection of problems from the textbook [?] on ordinary differential equations has been designed. The teachers found that the resulting interactive problems are useful from the didactic point of view, the use of the tool is intuitive and simple, and they could not have developed anything similar without ConsMath. The teachers have also warned us that before using the system on a larger scale with less advanced users like students, the behaviour of the editor of math formulae should be refined. Since this editor is a third-party component, we are planning to replace it by our own equation editor in a future release.

The rest of the paper is structured as follows: in the next section we shall describe from users' perspective the use of ConsMath, i.e. teachers' point of view in the process of creating a model for the interactivity of these applications by means of specifications by demonstration, and students' point of view when using these models. We will do it by illustrating a relevant example. The following section will present a more detailed view of the system. Finally, conclusions will be presented about this work.

2 Using ConsMath

When working with ConsMath, students have to solve problems of Mathematics by answering the questions asked by the system. Fig. 1 shows the resolution of a problem for the computation of a limit, as it appears in the initial static document. Once the interactive problem is designed with ConsMath, the student can solve it. Then the statement is shown to him, and he has to answer the questions that are posed to him one after another. Fig. 2 shows the first question the system poses the student while solving the problem from Fig. 1.

As an example of the dialog that can take place between a student and the system, if the student decides that the first step in the resolution of the problem in Fig. 1 is substituting $\sin x$ by x in the numerator, since they are equivalent infinitesimals, the system can show the student a simple case, like the

computation of the limit of $(x - \sin x)/x^2$; this method obviously does not work. ConsMath can show the student the correct resolution of this simplified problem together with the incorrect one, both of which are generated automatically on the bases of the work done by the designer. After this, the student is asked again about the resolution of the original problem.

We shall describe now the way in which the designer can specify the dialog that takes place between the student and the system when in tutoring mode. When the designer starts his work with the initial static document, the resolution of the problem disappears and the statement of the problem stays in the screen. He can then add interactive components like the selection buttons in Fig. 2. The designer then specifies the actions to be taken when the student makes a selection by clicking on one of the possible choices; at this point the buttons disappear, the remaining part of the resolution of the problem appears again, and the designer can use it as part of the text to be shown to the student afterwards. He can also keep showing selection buttons or input fields until the problem is solved.

In order to specify the dialog that takes place in case the student makes a different choice in the first step, the designer can use the left arrow that appears in the upper part of the design window, which allows him to turn back step by step to the same situation shown in Fig. 2. Then he can click on another button and start again the corresponding specification. The same can be done at each step of the resolution.

The limit in Fig. 1 can be computed in several ways, like using L'Hôpital's rule or substituting the functions that appear by their Taylor expansion up to some degree. ConsMath, that can generate the problem randomly, is able to notice while the student is working which the possible methods of resolution are, and adapt accordingly its reaction to the answers given by the student. For example, in the case shown in Fig. 2, only the second, fourth, fifth and last possibilities are accepted. The fifth possibility gives rise to more questions that finally lead to a deadlock where the student has to recognize that the problem can not be solved, while the other possibilities can give rise to successful resolutions of the problem. Problem statements are generated randomly from statement patterns. These patterns include generalized formulae like the following one, which generalizes the formula in the problem in Fig. 1:

$$\lim_{x \rightarrow 0} \frac{A + Bx + C \sin x + Dx^2 + Ex \sin x + F \sin^2 x}{x^m \sin^n x} \quad (1)$$

where A, B, C, D, E, F, m and n can have arbitrary values within some bounds fixed by the designer. Many times the designer imposes other conditions on the initial generalized formulae. In the previous example, the designer can ask naturally for the existence of two coefficients in the numerator that are not null. In this case, when building a new specific statement, the coefficients are generated randomly one or more times until the desired condition is satisfied. ConsMath includes generators of random numbers within given intervals (integer or rational), and generators of random mathematical expressions of specific forms like polynomials or simple trigonometric functions. For example, in the previous

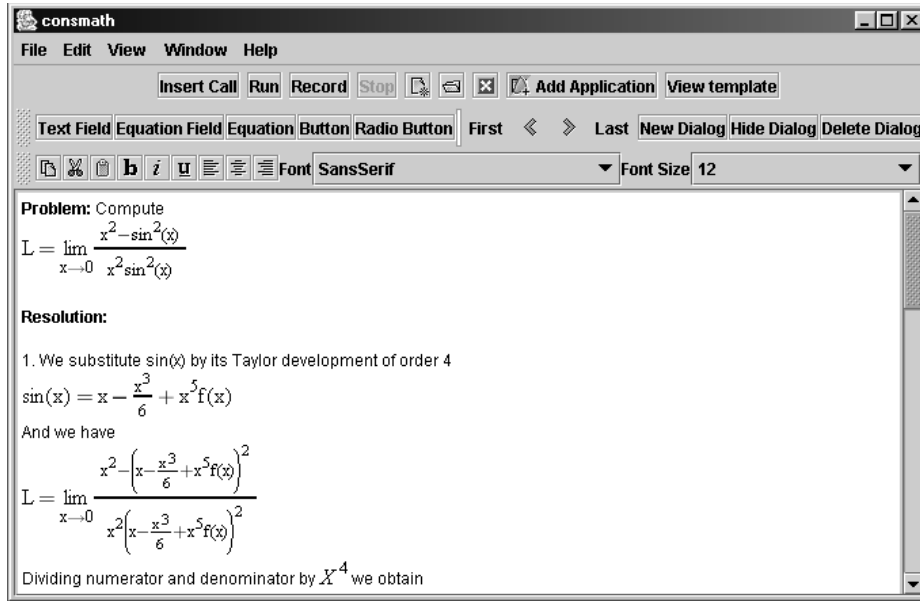


Fig. 1. Problem statement

pattern the numerator might also have been defined as a polynomial of degree 2 in x and $\sin x$ with the condition that it is not a monomial.

During the design of a problem, designers can alternate the way each formulae is shown between the specific formulae and the corresponding generalized one. This can be done by clicking on the formula with the right button of the mouse and making the choice they want. If the formula is not generalized, the generalized view will show the same information as the specific one. By editing the generalized formulae, designers can design them, as well as they can see the effect of their design on a specific example.

When a formula depends on another one, like when substituting $\sin x$ by its Taylor expansion,

$$x - \frac{x^3}{6} + x^5 f(x) \quad (2)$$

ConsMath allows designers to propagate the generalization from the original formula to the other one by means of a spreadsheet mechanism based on the use of constraints among parts of formulae. For example, the numerator of the third formula in Fig. 1 can be generalized to

$$A + Bx + C\left(x - \frac{x^3}{6} + x^5 f(x)\right) + Dx^2 + Ex\left(x - \frac{x^3}{6} + x^5 f(x)\right) + F\left(x - \frac{x^3}{6} + x^5 f(x)\right)^2 \quad (3)$$

and its value is substituted automatically in it. The same effect can also be obtained by giving the function within the limit in formula ?? a name like Q , and substituting $\sin x$ by formula ?? instead of the quotient in the third formula in Fig. 1. In this expression, the arrow indicates that a substitution has to be made. This can be done by just editing the generalized formula as indicated in the previous paragraph.

As said before, each time a pattern is defined in order to generalize an initial formula, the designer can specify a condition that must be satisfied. This can be done by typing a condition in the corresponding input field that depends on the variables defined in the previous generalization process. Similarly, each time the designer specifies the behaviour of the system when the student makes a choice, he can specify a condition for its application. This can be done in the same way. Hence, for example, the designer can specify that the first choice in Fig. 1 can be accepted in case $A=B=D=0$ by just typing this condition on the corresponding input field.

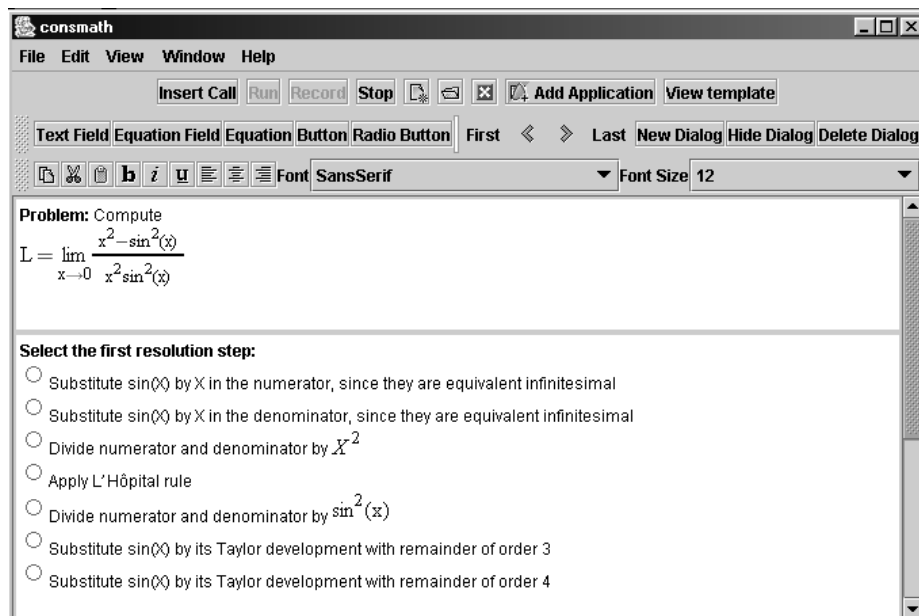


Fig. 2. Choice during problem resolution

Analogously, designers can specify that the questions to be asked to the students when solving problems depend on the specific problem that is being solved. For example, in case $A=B=D=0$ in the above generalized problem, it makes sense to ask the student if he wants to factorize $\sin x$ in the numerator. This is also specified by means of conditions attached to the multiple choice buttons that show the different choices that are available in general. From this point of view,

ConsMath is an adaptive system that admits different ways of resolution and reacts in different ways to the actions of the students depending on the specific problems that are being solved. Although a lot of work has been done in the last years in adaptive tutoring systems, none of them depends on deep properties of mathematical formulae and they have many limitations from the point of view of the interactive specification of the problems without the need of any programming.

3 ConsMath architecture and model

ConsMath consists of several java applications that are distributed according to a client-server architecture (Fig. 3). It allows designers to create courses as described in the previous section and, besides, both the execution and the design of these courses can be done in a collaborative manner.

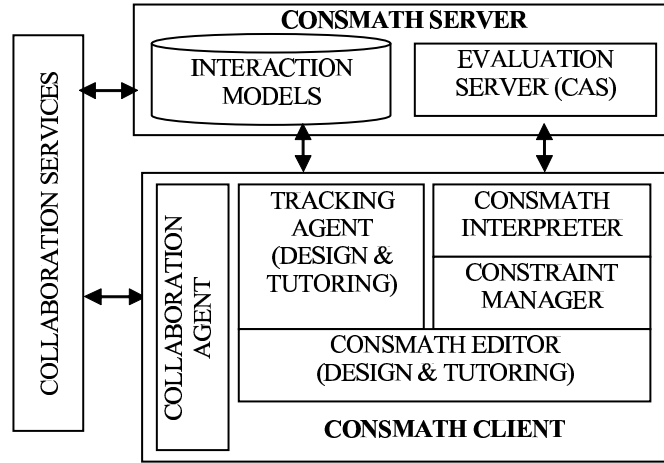


Fig. 3. ConsMath Architecture

In order to clarify the behaviour of ConsMath components, we will firstly describe each of them. These components are the following ones:

- An *editor of mathematical texts*, which supports symbolic relations among its components, and the specification of the interactivity using a programming by demonstration approach. This editor has two execution modes: the design mode (used by teachers), and the tutoring mode (used by students), which does not provide all the functionality of the design mode.
- The *ConsMath server*, which has a twofold functionality. Firstly, it serves as a central repository of courses or interaction templates, as courses are

saved persistently creating a library, and secondly it provides mathematical expression evaluation capabilities. This second functionality is performed by using a CAS (currently Mathematica), so that that ConsMath Server acts as a bridge between the CAS and the clients. As a consequence of these two functionalities, the ConsMath server simplifies the maintenance of the system and reduces costs, given that the clients just need the existence of one CAS running, which is at the server side, and the courses do not have to be replicated at each student's machine. The ConsMath server communicates with each client using Java Remote Method Invocation (Java RMI).

- The *tracking agent* is the component in charge of writing the interaction models during the design phase, and of interpreting these models in the tutoring phase.
- The *constraint manager* is the component that keeps all the dependencies in the document updated, sending to the ConsMath interpreter the expressions to be evaluated. Consequently, parts of these expressions need to be evaluated by the CAS, in which case the interpreter sends them to the ConsMath server for their evaluation.
- The *collaboration agent* is the ConsMath component which communicates with the collaboration services included in FACT, the collaboration framework used by ConsMath [?], in order to provide collaborative tutoring and designing support. The collaboration services are used by ConsMath users to participate collaboratively in the reviewing process as teachers or in the solving process as students. FACT is also written in Java and has a Client-Server architecture that communicates with the collaboration agent of each ConsMath client using Java RMI. FACT plays two main roles, firstly, it is a structured repository of collaborative sessions, which allows the users of the system, students or tutor, to collaborate in a problem resolution or review others work asynchronously, and secondly, it manages the synchronous collaborative sessions, so the users of each synchronous session share the same information and the same state of the ConsMath clients.

ConsMath users interact most of the times with the ConsMath editor. This editor can be used both by designers to create an interaction model and by students to execute this model. What designers do is to simulate user's actions and the answer of the system to these actions, by using programming by example techniques as described in the previous section. The tracking agent interprets these actions at design time to create the interaction model. When the editor is in tutoring mode, the agent listens to the actions of the student to find the answer that must be given, according to the interaction model, and reproduces this answer. In the rest of this section we will describe the document model of the ConsMath Editor, and the interaction model.

3.1 ConsMath Document model

In order to simplify the interaction, ConsMath documents can be divided in sub-documents, which can be hidden or locked to prevent the interaction of students with some parts of the document when in tutoring mode.

Each subdocument is formed by HTML text combined with components that represent its non-textual part. The components that form ConsMath documents can show formulae that correspond to MathML expressions or graphical functions. These components can be editors or browsers. There are also controls for the input of MathML expressions, multiple choice controls, buttons, and list boxes.

The ConsMath's constraint manager can relate the properties of the components. For example, ConsMath allows the construction of graphics that are updated automatically depending on some formulae that appear in the document. The constraint manager allows the specification of the properties of a component as ConsMath expressions, just like in a spreadsheet. Properties of components can be values or boolean conditions, like a property that specifies if a component is visible or not. On the other hand, the expressions that can define a property can include functions to give a name to some part of the expression, creating a variable, functions that evaluate an expression, or that incorporate the value of another variable in the expression, or functions that generate a mathematical expression randomly, with an associated pattern and a condition that the expression must satisfy.

The constraint system is also responsible for the automatic conversion as needed between different types of expressions (boolean, MathML, Mathematica or OOCSP, [?] properties). For example, the visible property of a component can be associated to a condition to indicate when that component must be shown. This condition can be written in MathML with embedded ConsMath functions. It can be specified using a WYSIWYG content MathML editor.

3.2 Interaction model

The interaction model is represented by a tree where each path, from the root to each leaf of the tree, defines an abstract interaction sequence. Final interaction sequences can be more complex, because the abstract sequence can have cycles and calls to other interaction sequences, creating complex interaction models.

This model is written by the tracking agent in the design phase creating a tree with two different interlaced zones, decision zones and performance zones. Decision zones are sub-trees with information to discriminate the different situations that can be produced by the student actions. This information is acquired by the tracking agent when listening to the actions of the designer when simulating the situations that can be produced later during the execution of the model by the students. For example, the designer can simulate that the student introduces the correct answer, later the designer goes back to the initial situation by using the left arrow that appears into the upper part of the design window, as shown in figure 2, according to the explanation given in section 2, and introduces an invalid answer, creating a simple decision tree with only two alternatives.

Each leaf of a decision tree is followed by a sequence of actions, in the performance zone, that correspond to the changes in the ConsMath Document that will take place after the situation described in the previous path of the decision

tree is detected. The information of the actions in the performance zones is acquired by the tracking agent by listening to the actions of the designer when in performance state. Also, performance zones can contain calls to other interaction models, or jumps to other points in the interaction tree.

In order to create the interaction model the designer navigates through the model going backward and forward creating new alternatives in a decision tree or creating new performance sequences for a particular decision path, etc. The tracking agent follows the designer actions, creating new nodes of information in the tree that represents the interaction model. A typical decision tree is formed by a sequence of events for each alternative. At run-time, when the tracking agent is listening to the events produced by the interaction of students with the current document model, and the agent recognizes one of the sequences, it emits the corresponding answer that is codified in the performance zone of the interaction tree. The answer usually consists of a new sub-document of ConsMath, with a dialog, and another decision tree, or a call to another interaction model, which usually corresponds to a subproblem call.

Also, the interaction model supports the combination, in any order, of different types of decision trees, forming more complex decision trees. These types are:

- *Sequence* decision tree: Is a tree where a path is matched when all events in the path take place in the order described by it.
- *And* decision tree: Is similar to the sequence decision, but the events can take place in any order.
- *Or* decision tree: In this case only one of the events described in a path of the decision tree is necessary in order to match that path.

The techniques described in this section allow the construction of a system as described in section 2.

4 Acknowledgements

The work described in this paper is part of the Ensenada and Arcadia projects, funded by the National Plan of Research and Development of Spain, projects TIC 2001-0685-C02-01 and TIC2002-01948 respectively.

References

1. Alfonseca, M., Lara, J.: Simulating evolutionary agent colonies with OOCSMP. In 17th ACM Symposium on Applied Computing (SAC'2002), track on AI and Computational Logic, Madrid (2002) 11-15
2. Asperti, A., Padovani, L., Coen, C. S., Guido, F., Schena, I.: Mathematical Knowledge Management in HELM. In MKM 2001 (First International Workshop on Mathematical Knowledge Management), RISC, Schloss Hagenberg (2001)
3. Cypher, A.: Watch what I do. Programming by Demonstration. MIT Press, Cambridge, MA (1993)

4. Diez, F., Moriyon, R.: Solving Mathematical Exercises that Involve Symbolic Computations. *Computing in Science and Engineering* 6 1 (2004) 81-84
5. Mora, M., A., Moriyón, R., Saiz, F.: Building Mathematics Learning Applications by Means of ConsMath. In *Proceedings of IEEE Frontiers in Education Conference*. Boulder, CO (2003) F3F1-F3F6
6. Mora, M., A., Moriyón, R., Saiz, F.: Developing applications with a framework for the analysis of the learning process and collaborative tutoring. *International Journal Cont. Engineering Education and Lifelong Learning* 13 3/4 (2003) 268-279
7. Quinney, D.: Computer-Based Diagnostic Testing and Student Centered Support. In Cliff Beevers, (ed.): *Computer-Aided Assessment in Mathematics Series*. Learning and Teaching Support Network (2001)
8. Simmons, G. F.: *Differential equations: with applications and historical notes*. McGraw-Hill (1981)
9. Sorgatz, A., Hillebrand, R.: MuPAD. In *Linux Magazin* 12/95 (1995)
10. Wolfram, S.: *The Mathematica Book* (fourth edition). Cambridge University Press (1999)
11. World Wide Web Consortium (W3C): W3C Semantic Web. URL: <http://www.w3.org/2001/sw/>.
12. World Wide Web Consortium (W3C): The MathML Language. URL: <http://www.w3.org/Math/>.