Dialog Issues for a Tutor System Incorporating Expert Problem Solvers

Erica Melis and Helmut Horacek
DFKI Saarbrücken and Universität des Saarlandes

Abstract

The training of operational skills in the domain under consideration is a key issue for tutor systems. This requires an elaborate presentation of a wide range of examples and associated explanations. Since preparing a documentation of many examples in sufficient variability and detail is an extremely time consuming task and therefore we propose the integration of expert problem solving systems (EPSs) that provide the technical competence for solving and demonstrating examples. Moreover, the integration of EPSs provide a basis for exploratory problem solving. In this paper, we elaborate the use of some EPSs integrated into the ActiveMath environment. This environment dynamically and adaptively generates an interactive mathematical document. When working with the interactive document, the EPSs can be employed for computing a solution, checking the user's solutions or for interactively solving a problem. In particular, for the latter elaborate dialogs are needed. In this paper, we characterize the role of EPSs from the point of view of an educational system, and we discuss the enhancements in the tutorial dialogs.

Introduction

The training of operational skills in the domain under consideration is a key issue for educational systems. This requires an elaborate presentation of a wide range of examples and associated explanations. However, preparing a documentation of examples in sufficient variability and detail is an extremely time consuming task.

We advocate in favor of employing expert problem solving systems (EPSs) into educational systems for several reasons. First, automatically working EPSs can find the solutions for problems which are not precomputed as discussed in the last paragraph. That is, the integration of EPSs can make sense from the course generation point of view because they produce (parts of) solutions on-line as opposed to the prefabricated and stored solutions in today's typical tutor systems.

Second, we want to support exploratory problem solving as a teaching strategy in tutor systems (Melis & Fiedler 2000). For this purpose automatically working EPSs can check the inputs of the user, e.g., a computer algebra system can compute a result that can be compared with the user's input; an automatic EPS can as well compute a partial solution that corresponds to a minor routine task for the user and thereby help to focus the user on the important tasks in learning. Moreover, an interactively working EPS can be used directly as a tool the user can construct a solution with. For a discussion see (Melis & Fiedler 2000).

One domain, where we believe that this approach is beneficial is mathematics. In particular, the course level that includes proofs of theorems and similar problem solving to some extend requires to learn in an exploratory way, i.e., not guided along a prefabricated solution. Prefabrication would demand to encode in advance the solutions or proofs for problems the user is able to solve.

Therefore, we designed the web-based ActiveMath environment for dynamically generating interactive documents (for learning mathematics) (Melis 2000). ActiveMath integrates several EPSs for mathematical problem solving and dynamically and adaptively generates an interactive mathematical document ID, in particular an algebra textbook at the undergraduate level.

Currently, we are elaborating the use of EPSs in this environment. According to their functionalities, ID will provide a variety of presentation and dialog features.

In this paper we first describe the ActiveMath environment including the EPSs. Then we characterize the role of EPSs in our environment. Finally, we address dialog enhancements in connection with the use of EPSs.

Preliminaries: Proof Planning

Proof planning builds an abstract representation of a proof (or problem solution) by constructing a sequence of plan operators that transforms a set of proof assumptions into a set of formulas containing the conjecture to be proved. The abstract plan operators are called methods and encapsulate fundamental (mathematical) proof techniques such as diagonalization and induction, as well as specific ones, such as a variety of estimation methods. A proof planning problem is defined by an initial state specified by the proof assumptions and the goal g given by the theorem to be proved. Roughly, an automated planner searches backward for an instantiated method M whose application proves a goal g and introduces M into the plan. The subgoals needed for the application of M replace g in the planning state. The planner continues to search for methods applicable to a subgoal and terminates if no open goals are left or if no further method can be applied. In some places the proof planning functionality requires the complementation by functionalities of domain-specific computational facilities such as computer algebra systems and constraint solvers. For instance, a constraint solver can collect all the constraints that a variable should satisfy for a proof, check for their consistency, and finally find a consistent instantiation satisfying all constraints.

When the proof planner is used interactively, the user can select the next goal to work on, select a method to be applied next from a library of proof methods, can appropriately instantiate a method, and combine it with others to perform parts of a proof and, ultimately, the entire proof to a given problem. Actually, the teaching and training of skillful choice and use of methods provides an excellent environment to acquire operational skills in more advanced mathematics. The control knowledge used to guide the search in automatic proof planning is represented by control rules. These can be communicated to the user as a basis for certain meta-cognitive activities.

Our Environment: ActiveMath

Our 'Interactive Textbook' project develops the ActiveMath environment which provides a web-based architecture as displayed in Figure 1, the basic knowledge representations, and techniques for second-generation interactive mathematics documents (textbooks, courses, tutorials) exhibiting, among others, the following features: separation of knowledge representation and functionalities, knowledge representation and communication in `OMDoc`-format which is an `XML` format that extends `OpenMath`, and adaptivity (Melis & Siekmann 1999a). The ActiveMath environment includes several computer algebra systems and an interactive proof planning system.

As summarized in Figure 1, the distributed web-architecture connects storages, presentation facilities, and mathematical services. Each stand-alone mathematical service needs a formal input in its input language which is produced by a communication shell wrapped around the service. Currently, ActiveMath integrates a mathematical knowledge base, MBase (Kohlhase & Franke 2000), a presentation planner, a user model, a pedagogical module, a session manager, and several EPSs (at the server side) and a web-server and browser at the client side.

From a unified formal and informal representation of mathematical content in MBase which is annotated with meta-data and dependent on the user model information and on pedagogical rules, the presentation planner creates an instructional graph (and its user-adaptive updating) of pointers to `OMDoc`-items to be presented to the user by the browser in `HTML`-format including pictures, buttons for calling Ωmega etc. The session manager fetches the actual content of these pointers from MBase. The session manager communicates the user's actions to the user history. It passes the user's requests for the proof planner Ωmega and for computer algebra systems (CASs) to the respective systems. Moreover, the session manager administers the current session and handles simple requests such as navigation, login, or changing display preferences. More complex tasks concerning document creation and modification are passed to the presentation planner.

The generated interactive document ID can have a button for using an EPS in case this was ruled pedagogically appropriate in the context of a particular example or exercise. Several EPSs can occur here, the computer algebra systems Maple (Char *et al.* 1986) and gap (Schönert 1995), and the proof planner Ωmega (Melis & Siekmann 1999b). The services of the former are common place. The service of the proof planner will be explained briefly.

The automatic proof planner computes (complete) proof plans that can be displayed to the user in a multi-modal way (see Figure 2) including a natural language verbalization (not displayed in the screen shot). The interactive proof planner uses the method- and control-knowledge acquired for automated proof planning to support active learning of mathematics and to provide feedback to the user. This feedback does not amount to a tight tutorial guidance but rather to a scaffolding in a more exploratory and active problem solving.

The Role of Expert Problem Solvers

Expert problem solving systems can play the following roles in an educational system:

- they can provide complete solutions to a problem or subproblem in the curriculum, e.g., the result of a simplification, differentiation, or a proof plan
- they can be used to check the correctness of a user input in case the correct solution/input is not predetermined
- interactive EPSs can be used for a user-guided interactive problem solving.

In order to accomodate EPSs in an educational system and to set up techniques for communication about their use, the tutor system requires a model of each EPS, comprising:

- a conceptualization of the generic functionality of the EPS that can be communicated to the user,
- a communication model for specifications and results of the EPS, in terms suitable to the domain model of the tutor system.

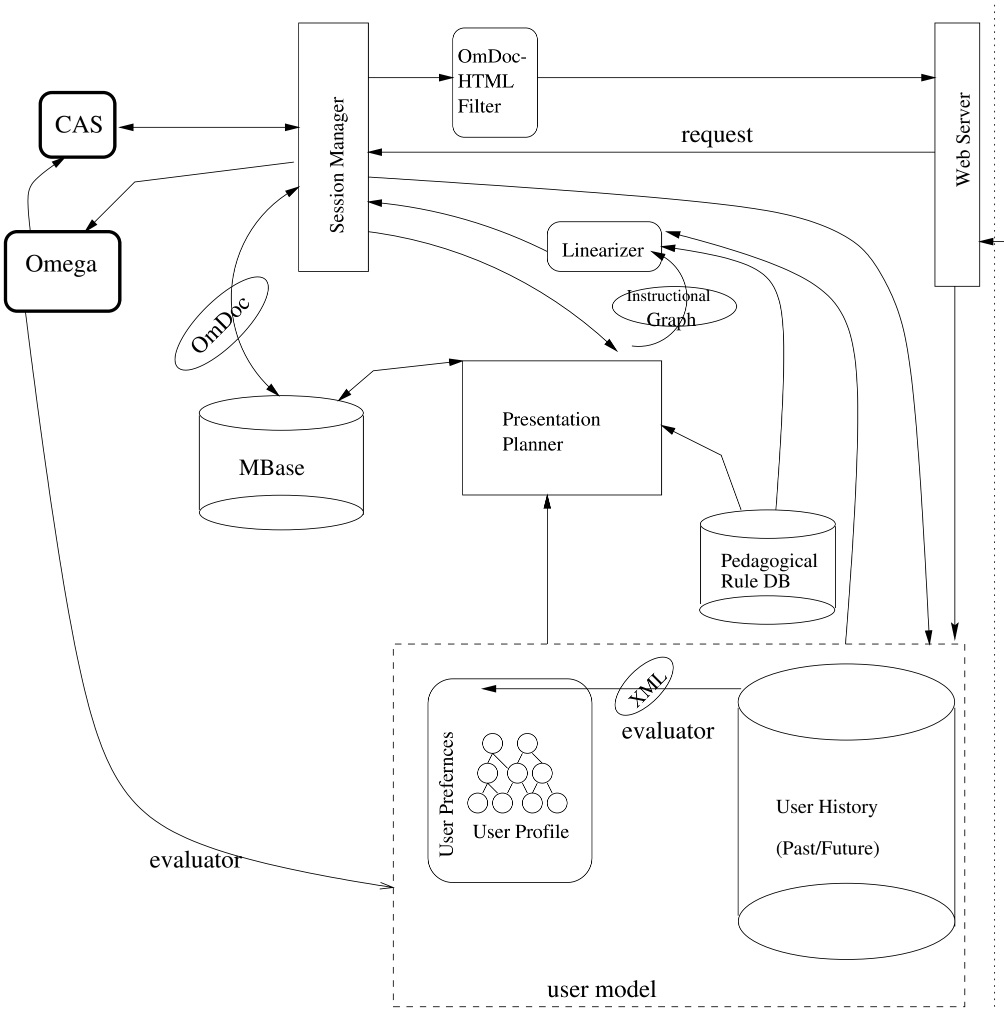

Figure 1: Architecture of ActiveMath

In order to address this issue systematically, we distinguish several categories of systems based on degrees of specification complexity, accessibility of their results, and communication about their use (the first category constitutes the simplest case, and the remaining ones are complementary extensions that can, in principle, be combined with one another):

- *Black-box systems* The model of systems in this category is restricted to mere functionality. Examples in our domain are computer algebra systems and some constraint solvers.
- *Controled systems* The use of these systems manifests itself in the provision of control parameters in addition to mere input specifications. An example in our domain is an automated proof planner with different options to choose a set of methods and control heuristics.
- *Informing systems* The use of these system manifests itself in the production of detailed descriptions which go beyond the mere specifications of results. Examples are traditional automated theorem provers which provide proofs rather than an yes/no output and an automated proof planner with a documentation of a resulting proof and its current state.
- *Interactive systems* The use of these systems, finally, manifests itself in additional communication during processing. An example in our environment is the interactive proof planner which allows the provision of specifications in a dynamic way dependent on the proof and user environment.

Consider, as an example for black-box systems, a computer algebra system whose functionality is captured as algebraic term manipulation, which is restricted according to the coverage of the specific system. Communication with a computer algebra system requires pointing facilities to a given formula that is subject to manipulation and goal term structure specification tools including reference to conceptualizations of uses.

Consider, as an example for an informing system, a traditional automated theorem prover such as OTTER (McCune 1990). Here the elaboration of the communication facilities may turn into a more labor-intensive task. Especially the nature of the items to be presented may require presentation facilities which have to perform considerable modifications to the content of the EPS's output. In particular, this concerns inference structures, where the needs of men and machine to understand them differ significantly. For machine internal uses, the underlying reasoning is represented entirely explicitly, in order to enable sound logical reasoning. In contrast, humans prefer partially implicit conveyance of inference-rich information, thereby exploiting the contextual setting and background knowledge to mentally reconstruct the underlying situation. Therefore, we have facilities that can reorganize inference structures, especially in varying degrees of explicitness motivated by psychological insights (Johnson-Laird & Byrne 1990). The method is applicable to general discourse and elaborated for mathematical proofs (Horacek 1999), where the decisions about leaving information implicit or not are based on the given environment and assumptions about the user.

Consider, as an example for an interactive system, the proof planner of ΩMEGA. Its functionality has been described in section . This functionality can be used in automatic or interactive mode. In interactive mode supporting the user is desirable. In both modes, the (partial) proof plan is subject to communication. One of our presentation facilities (Melis & Leron 1999) presents single methods as well as plans in hypertext. In the interactive mode additionally, choices, requests, actions, and questions of the user have to be communicated as well as the systems requests, clarifying questions, denial, and actions (see section .

Despite the integration and presentation effort, incorporating an EPS is useful if the specific information provided (be it the mere functionality in terms of input specifications and results or richer forms in terms of interaction) is beneficial for learning. Even a pure black box system proves beneficial, if the way the result is obtained is of neglectable importance for the user and the context of the tutorial system provides enough information for communicating about the use of the black box system in the particular situation. For example, when using a computer algebra system, for many users it is more important to learn about functionalities and motivations for its use rather than about details of its internal operations.

In addition to this basic form of integration, a tighter incorporation of an EPS in the overall context of the educational system may render additional benefits. This measure requires conceptualizations of specific uses of that EPS within the educational system. These conceptualizations may serve as descriptions for the specific use of an EPS (when derived from the immediately preceding goal) or as motivations for the use of that system in the overall context (when derived from a higher-level goal). For example, specific uses of a computer algebra system from other parts in the tutorial system include term simplification and factorization, the latter being a specific form of goal term specification. In both cases, the associated motivation is rendering the application conditions for some method feasible, through recasting the problem (in the case of term simplification) or through enabling splitting it into managable subgoals (in the case of factorization).

In the following, we demonstrate the use of EPSs and the associated dialog acts when these systems are embedded into a learning environment. The two exemplary systems used are a computer algebra system (category black-box) and an interactive proof planner (categories informing and interactive).

Dialog Acts

In ID, dialogs occur not only when an EPS is used but also outside the proper communication with EPSs, e.g., when a concept is searched for in a dictionary, when the user requests an advice for appropriate navigation, for the next actions, or for a repetition. Dialogs may also occur in the presentation of completed predetermined solutions.

In order to keep the communication task managable we currently use restricted flexibility in communication. On behalf of the user, specifications can be expressed by choosing from contextually updated menus and dialog boxes, while system presentations are composed of portions of freely generated natural language, combined with fixed text templates, which is a quite up-to-date technique in application-oriented natural language generation.

In this section, we first address issues of organizing communication. Then we briefly comment on the dialog acts used. Finally, we discuss communication specificities of the interactive proof planner, and we exemplify its use, including applications of a computer algebra system, through an elaborate example dialog that illustrates the envisioned functionality of our system.

At the beginning of a session, the user is asked to identify her degree of expertise in thematic areas and presentation preferences. This is done by offering the user a selection of thematic subareas in which she is invited to express her degrees of expertise, which may be one of 'none', 'partial', and 'full'. If 'partial' is chosen, a more pronounced subdivision of that area is offered as an option to make more precise specifications. In addition, some preferences can be expressed including the choice between formulas and natural language for presentations where this alternative is suitably applicable, the annotation with priovate notes, and more implicit or more explicit presentations.

In a learning session, the user is always allowed to select elements from a dictionary and to ask for information about the item indicated. Whenever a user request concerns an area in which she has previously expressed competence or, conversely, when she does not use an expected clarification option if a partial presentation does not address all aspects assumed to be unknown to the user, a system inquiry is potentially useful. In case of a missing clarification option, the system checks the user's understanding because it is very likely that the user has missed some point. In case of an informatory request on behalf of the user, related system requests are only made opportunistically due to the expected high frequency of this case. This behavior is chosen because the degree of precision of the system's evidence is low and the user may ask for this information not only in case of unfamiliarity.

Categories of Dialog Acts with EPSs

Common dialog acts, such as requests for confirmation, requests for action, confirmations, denials, and actions (including the provision of information) occur in our environment, too, sometimes treated in a finer-grained specificity. Hence, the relevant types of dialog acts in our educational environment include requests for navigation, request for presentation, requests for concerning domain knowledge, responses, actions, motivations, and questions of the following variety

- What? This is a terminological question asking for a term explanation.
- Why? In case of a successful operation, this asks for the motivation behind its use, which is derivable from the conceptualization of the use of an EPS, either from the internal connections within an interactive EPS or via that EPS which refers to the EPS under consideration. In case of an unsuccessful operation, this amounts to communicating an error message. Subsequent repetition of that question triggers more detailed answers.
- How? This request addresses the internal structure of the functionality of an EPS, provided this is accessible. This applies, for example, to references to another EPS from the EPS under consideration. Subsequent repetitions of this request recursively apply to the functionality of an EPS referred to this way. This request is not applicable to black-box systems.

In order to set up the proper communication with an EPS of whatever category, the following dialog acts are needed:

- motivating the use of an EPS,
- input of problem and data.
- conveying the responses from EPS, including the content of error messages.

For systems other than black box systems, there are additional dialog acts:

- presentation of the alternative methods/functions of an external system (interactive systems),
- user requests for clarification,
- explanatory presentations of result obtained by an EPS (informing systems).

Dialogs with the Interactive Proof Planner

Possible (multi-modal) dialogs for communicating with the interactive proof planner are similar to the dialogs necessary for general mixed-initiative planners, where the user and the expert problem solving system can take the initiative to jointly solve a problem. These dialogs comprise a wide range of communication acts including the submission of a problem by the user (solve X), or by the system (conjecture X), direct requests (show me the plan, help request), questions (which object satisfies property X), assertions (take object a rather than b), suggestions (choose method Y), acceptances or rejections (okay, no), as well as more nonverbal acts such as selection from menu, pointing, and dragging screen objects or presenting an information graphically. A dialog management has to combine different modalities of user communication, e.g., (spoken) language, diagrams (pointing), button clicks. Similarly, different modalities can be used when the system communicates with the user.

In addition, dialog acts are necessary in order to explain the method itself and to explain which method has been choosen and why it has been chosen. Moreover, from the system's side context-sensitive commenting dialog acts are necessary to guide the user tightly (good/bad choice) or to support exploration or scaffolding (method not applicable).

The EPS that is by far the most important one in the following example is an interactive proof planner. Other EPS come into play only indirectly, through reference from that proof planner, so that the motivations for using them are accessible from the planner only. The proof planner's graphical user interface as well as ID provide access to a problem library. The user can choose between actions (choosing a problem and specifying the next operation to be performed) and asking questions about the action just performed by the system. These include what, why, and how questions.

The following elaborate dialog demonstrates the suitability of incorporating EPSs into an educational system. Individual utterances are labeled S for dialog contributions of the system and U for user requests and commands. The latter are only expressed in natural language text for the sake of readability, since they are in fact composed of dynamic menus enhanced by dialog boxes. In addition, the system utterances are numbered to enable a convenient reference to comments.

The example dialog is embedded in a session about solving a limit problem, and the user is supposed to be acquainted with the basic terms and theorems in this area of calculus such as the definition of limit, as well as with some sort of default proof operations (such as definition expansion) and to some extend with the methods (`Solve*` and `ComplexEstimate`.

S (1): Use the "examples" menu to specify an exercise.

U: (selects "lim-square" from the given set of alternatives – which means that the limit of x^2 is a^2, provided x goes to a)

S (2): Ok, specify a method.

U: Apply default techniques.

S (3): Ok, then it remains to show that $|x^2 - a^2| < e$ for some e. Specify the next operation.

U: Apply `Solve*`.

S (4): Applying `Solve*` fails.

U: Why?

S (5): The application condition for `Solve*` is not satisfied. Try `ComplexEstimate`.

U: Why is the application condition is not satisfied?

S (6): In order for the application condition to be satisfiable, the goal to be proved must be unifiable with a similar hypothesis. However, $|x^2 - a^2| < e$ is not unifiable with $|x - a| < d$

U: What is `ComplexEstimate`?

S (7): The method `ComplexEstimate` reexpresses a complex algebraic term as a decomposable expression made out of simpler terms. The estimation of the complex term can thus be reduced to subsequent estimations of these simpler terms which then become the new subgoals.

U: Why is `ComplexEstimate` promising?

S (8): The motivation for using the method `ComplexEstimate` is to break a complex term into parts for which it is hoped that estimating them is eased.

U: Apply `ComplexEstimate`.

S (9): Applying `ComplexEstimate` produces two new goals: $|x - a| < e/(2 * m)$ and $|x + a| \leq m$ for some real number m with $0 < m$.

U: How was that derived?

S (10): In general terms:
For an inequality $|term| < e$, the term $|term|$ is re-expressed by $|a * b + c|$. By the triangle inequality we have then $|a * b + c| \leq |a| * |b| + |c|$. If we could prove $|a| \leq m$, $|b| < e/(2*m)$, and $|c| < e/2$ for some $0 < m$, this would imply the truth of the original inequality.

U: How was that derived?

S (11): The re-expression is internally computed by a computer algebra system.

U: Apply `Solve*` to the new goal $|x - a| < e/(2 * m)$.

S (12): Okay. Applying `Solve*` yields the constraint $d \leq e/(2 * m)$ to be satisfied.

U: Apply `Solve*` to the goal $|x + a| < m$.

S (13): Applying `Solve*` fails.

U: Why?

S (14): $|x + a| < m$ is not unifiable with $|x - a| < d$.

U: Apply `ComplexEstimate` to that goal.

S (15): Applying `ComplexEstimate` yields the constraints $|1| \leq m1$, $d \leq m/(2 * m1)$, and $2 * |a| < m/2$ for some real number $m1 > 0$. The problem is now reduced to a set of constraints of d, m, and $m1$ for which an instantiation satisfying them can be found automatically.

U: Why?

S (16): This operation is internal to the system used for this purpose – a constraint solving system.

U: Show me the constraints.

S (17): The resulting constraints are $d \leq e/(2 * m)$, $d \leq m/(2 * m1)$, $4 * |a| < m$, $0 < e$, $0 < m$, $|a| \leq m$, and $1 \leq m1$.

U: Show me the proof.

S (18): (displays the proof graph of Figure 2)

U: Show me the verbal proof.

S (19): The problem is to show that the limit of x^2 is a^2, provided x goes to a. This means that $|x^2 - a^2| < e$ must hold for some e provided that $|x - a| < d$ holds for some d. The method `ComplexEstimate` is applied twice: (1) the inequality $|x^2 - a^2| < e$ holds provided that the constraint $d \leq e/(2 * m)$ and the inequality $|x + a| < m$ hold for some m with $0 \leq m$. (2) $|x + a| < m$, in turn, holds provided that the constraint $d \leq m/(2 * m1)$ holds for some $m1$ with $1 \leq m1$. The resulting constraints can be satisfied by $m1 = 1$, $m = 4 \cdot |a| + 2$, and $d = min(\epsilon/(8 \cdot |a| + 4), 2 \cdot |a| + 1)$, which proves the theorem.

At the beginning of the dialog, the system offers the user the way to acquire operational knowledge (1). After the user has made her selection, the system invites her to proceed (2). Then the user requests the system to proceed on its own first, which is a short-cut to focus on crucial techniques. The system reports then on the resulting goal to be proved by techniques other than the "default" ones (3). To accomplish this goal, the user attempts to apply a straightforward method, which the system finds to be unsuccessful (4). In answering the subsequent request, the system restricts itself to a mere generic answer (5), but adds a more detailed one including the instantiation (6) on the insisting request. Pursing the clarification dialog, the system provides a description of a method's functionality (7), followed by the underlying motivation (8).

The first progress towards solving the given problem is obtain by applying the operation suggested by the system and finally selected by the user, and this result is described in (9). The following clarification requests can only be answered as far as the description of the functionality of the method `ComplexEstimate` is concerned, including the instantiations of problem the method is actually applied to (10). Answering the subsequent request, which tends to address the internal functionality of a black box system, is restricted to naming the category of that system (11). The following

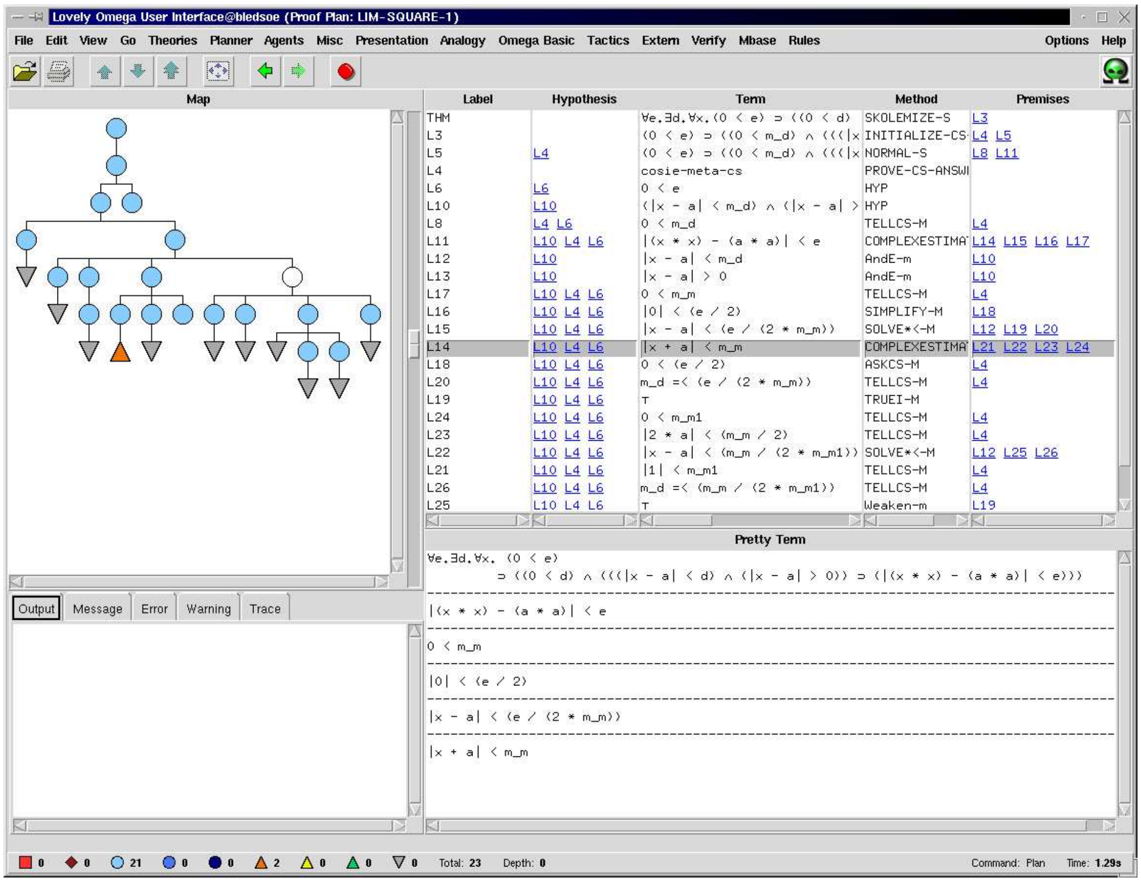

Figure 2: Screen shot for the completed proof of limsquare

portion of the dialog ((12) to (15)) is structurally similar to previous parts, except to the partial answer in (14), since the other parts of the answer were already given elsewhere ((5) and (6)). The dialog is concluded by demonstrating the variation in proof inspection tools the user has at her disposal.

It has to be noted that a particular benefit of using EPSs comes from the option in building examples of candidate theorems freely, for example through composing arithmetic expressions. In the limit problem chosen, the user could build arbitrary compositions out of functions for which the limit is known. In order to enable an interesting conversation about the use of EPSs, modeling the central ones as interactive systems is highly recommended, while treating supplementary systems with reduced modeling effort proved sufficient in our case.

Note, that the system explanations in this dialog do not refer to a conceptualization of the EPS used (the computer algebra system, because this is inaccessible to the system for a black-box EPS) but to the environment of its use (here, the context of the plan state and the related methods).

Conclusion

The paper addresses dialogs in a (mathematics) educational system that integrates expert problem solving systems such as computer algebra systems and a proof planner. This educational system has the facilities to dynamically generate solutions and to support exploratory learning. However, the integration of stand-alone systems has a price not only in terms of the integration itself but also with respect to the dialog facilities of the educational system since it creates the additional problem and opportunity of organizing a user dialog for the communication of each EPS.

We characterized different categories of EPSs and the corresponding classes of dialog acts. In particular, we looked at a computer algebra system which is a black box system and at the interactive proof planner which is an informing interactive system. The dialogs with the proof planner have been discussed in more detail and exemplified by a dialog fragment.

Through the use of these EPSs, the coverage of educational systems in terms of offering examples to train operational skills can be extended significantly.

References

Char, B.; Fee, G.; Geddes, K.; Gonnet, G.; and Monagan, M. 1986. A Tutorial Introduction to MAPLE. *Journal of Symbolic Computation* 2(2):179–200.

Horacek, H. 1999. Presenting proofs in a human-oriented way. In Ganzinger, H., ed., *Automated Deduction – CADE-16*, volume 1632 of *Lecture Notes in Artificial Intelligence*, 142–156. Springer.

Johnson-Laird, P., and Byrne, R. 1990. *Deduction*. Ablex Publishing.

Kohlhase, M., and Franke, A. 2000. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation* 11:1–37.

McCune, W. 1990. Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois.

Melis, E., and Fiedler, A. 2000. On the benefit of expert services in mathematics education systems. In Boulay, B. D., ed., *Proceedings of ITS-2000 workshop on Modeling Human Teaching Tactics and Strategies*, 84–85.

Melis, E., and Leron, U. 1999. A proof presentation suitable for teaching proofs. In Lajoie, S., and Vivet, M., eds., *9th International Conference on Artificial Intelligence in Education*, 483–490. Le Mans: IOS Press.

Melis, E., and Siekmann, J. 1999a. Vorprojekt: 'Interaktives Lehrbuch'. Project proposal.

Melis, E., and Siekmann, J. 1999b. Knowledge-based proof planning. *Artificial Intelligence* 115(1):65–105.

Melis, E. 2000. The 'interactive textbook' project. In *Proceedings of CADE-17 workshop on deduction in Education*, 26–34.

Schönert, M. 1995. *GAP – Groups, Algorithms, and Programming*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany.