Dialogs for a Tutor System Incorporating Expert Problem Solvers

Erica Melis and Helmut Horacek
DFKI Saarbrücken and Universität des Saarlandes

March 29, 2000

Abstract

The training of operational skills in the domain under consideration is key for tutor systems. This requires an elaborate presentation of a wide range of examples and associated explanations. Since preparing a documentation of examples in sufficient variability and detail is an extremely time consuming task, the integration of expert problem solving systems (EPSs) that provide the technical competence for solving and demonstrating examples seems to be a promising approach. This paper presents a categorization of EPSs that is based on degrees of specification complexity and accessibility of their results and elaborates the use of some systems embedded into the interactive mathematical textbook ID. We discuss some dialog tasks and problems that arise from the integration of those independent systems.

1 Introduction

The training of operational skills in the domain under consideration is key for tutor 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.

In order to partially automate the presentation of problem solving and in order to allow for exploratory problem solving as a teaching strategy in tutor systems [3] we advocate in favor of employing expert problem solving systems into tutorial systems.

The integration of such systems makes sense for tutor systems for several reasons, among them cognitive and pedagogical ones [3]. Likewise, 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.

However, this integration has a price not only in terms of the integration itself but also with respect to the dialog facilities of a tutor system since it creates the additional problem of organizing a user dialog for the communication of each EPS. For example, details of solutions or formal justifications may or may not be suitable to be presented to a particular user for didactical reasons. Therefore, the dialog with the EPS and the presentation of the solutions it produced should be generated in a user-adaptive manner from the output of that system.

One domain where we believe that this approach is beneficial is mathematics, which is also our domain of application. In particular, the course level that includes proofs of theorems and similar problem solving makes it extremely hard to encode stepwise solutions or proofs for problems the user is able to solve in an exploratory way, i.e., not guided along a prefabricated solution.

Therefore, we have designed the interactive mathematics textbook ID (Interactive Document)[4] at the undergraduate level so that it integrates several EPSs. According to the functionality of EPSs integrated, ID can provide a variety of presentation and dialog features.

In this paper we first describe the environment of our interactive mathematics textbook ID including the EPSs. A categorization of EPSs follows based on degrees of specification complexity and accessibility of their results. Then we discuss the implications of the EPSs' integration in our environment, in particular, implications for the associated dialog features.

2 Our Environment

Our project ID develops an interactive (mathematics) textbook for undergraduate mathematics that will support exploratory problem solving. It also investigates the web-based dynamic generation of an interactive document from a unified formal and informal representation of mathematical content that is annotated with meta-data as well as pointers to servlets [2].

In ID several EPSs are integrated, the computer algebra systems MAPLE [1] and GAP [7], the constraint solver COSIE [6], and the proof planner ΩMEGA [5]. The services of the former are common place. The service of the proof planner will be explained briefly.

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 formulae containing the conjecture to be proved. The abstract plan operators are called *methods* and encapsulate (mathematical) proof techniques such as diagonalization and induction. 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. A proof planner can be used interactively. In this case, it asks the user to choose the method to be introduced into the plan next.

The proof planner can support the learning of operational skills in theorem proving/problem solving. On the basis of a library of proof *methods* the user can select a method, appropriately instantiate a method, and combine it with others to perform parts of a proof and, ultimately, the entire proof to a given problem. Evidentially, the training of skillful choice and use of methods provides an excellent environment to acquire operational skills in more difficult mathematics.

In some places the proof planning functionality requires the complementation by functionalities of computational facilities such as computer algebra systems and constraint solvers.

The various facets?? of using?? method is subject to hints and explanations in the tutorial setting.

3 Categories of Expert Problem Solvers

In order to arrange embedding of EPS and setting up techniques for communication about their functionality, we distinguish several categories of systems (the first one constitutes the simplest case, and the remaining ones complementary extensions that can, in principle be combined with one another):

- *Black-box systems* The view of the tutorial system on systems of this category is restricted to mere functionality. Examples in our domain are computer algebra systems and certain uses of constraint solvers.

- *Controled systems* The use of these systems manifests itself in the provision of control parameters in addition to mere input specifications.

The typical example in our domain is an automated proof planner, with its various search options.

- *Informing systems* The use of these system manifests itself in the production of detailed descriptions which goes beyond the mere specifications of results. The typical example in our domain are all kinds of theorem provers (automated and interactive ones), with the trace documenting a proof.

- *Interactive systems* The use of these systems, finally, manifests itself in additional communication during processing. The typical example in our domain is an interactive theorem prover, which allows the provision of specifications in a dynamic way, in dependency of the proof state.

Incorporation of a system of any of these categories 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 the use within a tutorial session. Specifically, a pure black box system proves beneficial, if the way how the result is obtained is of neglectable importance, and the context of the tutorial system provides enough information for communicating about the use of black box system in the particular situation. For example, when using a computer algebra system, we believe that it is more interesting to learn about techniques and motivations for its use rather than about details of its internal operations.

In order for the embedding tutorial system to be able to play such a mediator role (and more than that), it requires a model of each EPS and associated tools in some complementary aspects:

- A conceptualization of the generic functionality of the EPS

- Interaction tools to handle specification provision for interactive systems

- Presentation facilities for data structures that constitute results

In addition, a tighter incorporation of an EPS in the overall context of the tutorial system requires:

- Conceptualizations of specific uses of the EPS from other parts of the tutorial system, if an appropriate description for the specific use is available

- Specifications to derive suitable higher-level goals as motivations for the use of the EPS in a given context

Consider, as a relatively simple example, computer algebra systems. Their functionality is captured as denoting some sort of algebraic term manipulation, restricted according to the coverage of the specific system or systems connected. Specific uses needed for 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 or through enabling splitting it into managable ones, respectively. Communication proper requires pointing facilities to a given formula that is subject to manipulation and goal term structure specification tools including reference to conceptualizations of uses.

In the following, we demonstrate the use of systems of several of these categories when they are embedded in a tutorial environment. We do not address control systems in our environment, because most of the associated parameters are highly technical and divergent from (human) domain concepts.

The two exemplary systems used are, a computer algebra system (category black-box) and an interactive proof planner (categories informing and interactive system). Thereby, we concentrate of the communication about the use of the EPS.

4 Dialog about the Use of External Systems

In order to keep the communication task managable we 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.

Categories of Dialog Contributions in ID In order to set up an environment for handling communication with EPS and keeping it, two kinds of dialog act pairs are used:

- stimulations for initializing and updating user model and associated responses

- requests by the user concerning domain knowledge in general and the system responses

The first and the second item are dialog acts to acquire environmental parameters for selecting among alternative presentation contents and tailoring them according to the contextually acquired evidence. The third category applies to interactive systems only, while the remaining categories concern all external systems, including failures in their use and user initiated extensions. We do not address control systems in our environment, because most of the associated parameters (e.g., for automated theorem provers) are highly technical and divergent from (human) domain concepts; an exception is the set of theorems or lemmas to be used within a proof, which is adopted according to the context of the tutorial session, and additionally required axioms are explicitly mentioned in presentations resulting from the use of this external system. Instead of asking the user for possible options, these systems are used with some defaults or motivated choices in case the tutorial system has adequate information about that, and the associated distinctions are hidden form the user.

4.1 Dialogs for Computer Algebra Systems

4.2 Dialogs forProof Planner

In the beginning of a session, the user is asked to identify his/her degrees of expertise in thematic areas and presentation preferences. This is done by offering the user a selection of subareas of the domain, in which he/she is invited to express his/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, which include the choice between formula and natural language for presentations where this alternative is suitably applicable, and more implicit or more explicit presentations. At any time the user is allowed to select elements from a layered index of domain concepts, to ask for information about the item indicated. Whenever a user request concerns an area in which he/she has previously expressed competence or, conversely, when he/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, since 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. In order to set up communcation with an EPS of whatever category, the following dialog acts are available:

- conveying the responses from EPS, including the content of error messages
- motivating the use of an EPS
- user requests for clarification

For interactive systems, there are two additional dialog acts:

- presentation of choice of alternative methods/functions of an external system
- explanatory presentations of result obtained by an EPS

In the follwing, we elaborate the use of these kinds of dialog acts in two exemplary EPS - computer algebra systems and interactive proof planners.

4.3 Computer Algebra Systems

4.4 Interactive Proof Planners

5 Conclusion

Dynamic curriculum, dynamic generation of solutions, combination of several modalities in dialogs.

References

[1] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, and M.B. Monagan. A Tutorial Introduction to MAPLE. *Journal of Symbolic Computation*, 2(2):179–200, 1986.

[2] The ΩMEGA group: J. Siekmann, Ch. Benzmüller, L. Cheikhrouhou, A. Fiedler, A. Franke, M. Kohlhase H. Horacek, A. Meier, E. Melis, M. Pollet, V. Sorge, C. Ullrich, and J. Zimmer. Adaptive course generation and presentation. In P. Brusilovski, editor, *Proceedings of ITS-2000 workshop on Adaptive and Intelligent Web-Based Education Systems*, Montreal, 2000.

[3] E. Melis and A. Fiedler. On the benefit of expert services in mathematics education systems. In B. Du Boulay, editor, *Proceedings of ITS-2000 workshop on Modeling Human Teaching Tactics and Strategies*, Montreal, 2000.

[4] E. Melis and J. Siekmann. Interaktives Lehrbuch. Project Proposal.

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

[6] E. Melis, J. Zimmer, and T. Müller. Integrating constraint solving into proof planning. In Ch. Ringeissen, editor, *Frontiers of Combining Systems, Third International Workshop, FroCoS'2000*, number 1794 in Lecture Notes on Artificial Intelligence, pages –. Springer, 2000.

[7] Martin Schönert et al. *GAP – Groups, Algorithms, and Programming.* Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 1995.