A High-Level Categorization of Explanations
A Case Study with a Tutoring System

Helmut Horacek

Universität des Saarlandes
F.R. 6.2 Informatik, Postfach 1150
D-66041 Saarbrücken, Germany
email: horacek@cs.uni-sb.de

Abstract. Giving explanations is an important functionality of most systems that perform some sort of problem-solving, including intelligent tutoring systems. In order to make explanation techniques transparent and better reusable across kinds of applications and domains, various categorizations have been proposed, but they appear to be on a comparably low level of abstraction, oriented on the surface form of explanatory requests. Inspired by our work on the tutoring system *Dialog* that teaches students how to prove mathematical theorems, we distinguish categories of explanations according to the depth of understanding that these explanations aim at, ranging from expositions of reasoning chains to ingredients of problem solving techniques. The modular design of our system does not only allow the identification of methods closely related to these categories, but it also enables us to clearly distinguish methods for problem solving from methods for proper explanation. We are convinced that the field as a whole can profit from factoring out *explanation methods per se*, which have much to do with presentation and tailoring issues, and methods for problem-solving that provide enhanced evidence about the results and support inquiries on top of them – the proper *explanation-aware problem-solving*.

1 Introduction

Giving explanations is an important ingredient of many systems that exhibit reasoning capabilities, including knowledge-based systems, question-answering systems, and intelligent tutoring systems. Categorizations of explanations are frequently oriented on types of requests, such as "Why are you doing <x>?", or "How did you avoid the typical constraint <p>?" [23]. The perspective in this categorization lies on selective content determination, which is intertwined with problem-solving methods and reasoning about their results. We feel that this approach is appealing for these kinds of explanations, but it is hard to generalize it for explanations that involve problem-solving on a larger scale, reasoning about the results obtained and the ways how they can be obtained.

We are convinced that the field as a whole can profit from factoring out *explanation methods per se*, which have much to do with presentation and tailoring issues, and methods for problem-solving that provide enhanced evidence about the results and support inquiries on top of them – the proper *explanation-aware problem-solving*. Such a separation of functionalities would obviously strengthen the reuse of methods across applications, and it is likely to make the use of problem-solving systems for subsequent explanatory requests more effective. This conception is inspired by the modular design of our tutoring system *Dialog* [4] that teaches students how to prove mathematical theorems. In this paper, we discuss how this separation of functionalities enables us to address categories of explanations according to the depth of understanding that these explanations aim at, ranging from expositions of reasoning chains to ingredients of problem solving techniques.

This paper is organized as follows. We first motivate our approach. Then we characterize our explanation categories in terms of functionalities and methods, with reference to their use in a variety of knowledge-based systems covering a number of domains. We give a brief description of the *Dialog* system, followed by describing its explanation-relevant techniques according to our categorization. Finally, we summarize our ingredients for categorizations of explanations.

2 Motivation and Categories of Explanations

Several categorizations of explanations have been proposed in the literature, most of them oriented on types of requests, including [30, 34, 23]. For example, the categorization underlying CALO [9, 23] comes with a catalog of methods for determining the content required for addressing these questions. In addition, a plan language has been developed and successively refined [24], which takes care of conveying responses of the problem-solving components to be used for tailoring the presentation of explanations. Some complementary approaches aim at producing explanations of a different kind, including justifications for problem-solving results exposing the complete reasoning task, such as in the project HALO [27], and reasoning about potentially flawed actions as compared to appropriate problem-solving behavior [11], which is typical for intelligent tutoring systems.

For studying approaches to giving explanations, intelligent tutoring systems are an excellent target, since they cover a variety of situations and purposes in which explanations play a certain role. Ultimately, the purpose of these systems is to teach the student problem-solving capabilities in the domain at stake, by eliciting partial contributions to addressing specific problems studied in a tutorial session. In doing this, intelligent tutoring systems cover the kind of explanations mentioned above, especially explanations concerning problem-solving performance.

For our categorization, we assume that explanations always refer to some sort of problem-solving which requires a non-trivial inference task and which either gives a unique solution, a set of solutions of equivalent value or with compensative properties, or at least a state in exploring the problem to a certain depth, which gives evidence about the state of affairs uncovered so far, in particular still promising paths to potential solutions and assessments about their state, such as a collection of competing arguments. The categorization is then oriented on a separation between

1) the solution per se,
2) inferences that do not contribute to the solution, and
3) evidence about how the solution can be uncovered in the given problem space, yielding:

- *Explanations aiming at the provision of justifications*
  This category of explanations addresses the problem solution per se. It aims at illustrating the reasoning process in terms of exposing the chain of inferences that proves the validity of the solution on the basis of the given context. Various perspectives on these inferences and adequate forms of condensation and aggregation can be adopted to built an adequate presentation out of these ingredients. The purpose of this explanation category is to make the reasoning behind a machine-found solution accessible to a human user, to make the justification given checkable to ensure a trust in its validity or at least plausible to obtain some confidence.
- *Explanations aiming at the support for understanding of the problem space*
  This category of explanations adresses the problem space, in particular solution attempts and somehow plausible inferences that have been tried out or may have done so, but did not lead to sufficient success. These ingredients may be compared to the established solution in ways that depend on the nature of the problem at hand, including the proof of falsity and numerical or qualitative preferences. The purpose of this explanation category is to make the assessment underlying a machine-found solution plausible to a human user, to make the preferences over alternatives checkable or at least plausible to ensure a trust in the superiority of the solution.
- *Explanations aiming at the development of performance capabilities*
  This category of explanations addresses techniques on how the problem space is explored, and how solutions can be established. Regarding the proper results, the associated explanations essentially address meta-information. The purpose of this explanation category is to enhance understanding in the problem-solving behavior of the system, which may be used to get acquainted with underlying assumptions and think about their impacts, or it may contribute to understanding the procedures applied or the principles underlying them, so that the user can develop skills to perform the underlying reasoning on his own.

3 Categories of Explanations

In this section, we describe technqiues for addressing the categories of explanations, as outlined in the previous section. To a certain extent, we put emphasis on tutoring systems, especially in sections 3.2 and 3.3 but, in general, we discuss this topic on a broader basis.

3.1 Explanations Aiming at Providing Justifications

Explanations in this category are mostly conceived as chains of inferences, when viewed in purely structural terms. Alternations and multiplicity of elements or subchains in the chains of inferences may be present, in case there are alternative ways to justify intermediate or final conclusions, or if there are multiple ways of reasoning leading to conclusions of a quality that makes them at least competitive. A prototypical example is the main variation produced by a chess program, as the heuristically justified "solution" to some position when computed with given limited resources. In the general case, the result of the reasoning process justifying the solution to some given problem manifests itself in a solution tree, which is typically developed as a proof graph by exploiting multiple references to subproblem solutions. A reasonable explanation of this kind of structure amounts to a presentation that interprets this structure as content specifications, transforms it into a suitable format, and accommodates it to the communicative needs of the audience; even if considering only content matters in this presentation, experience tells us that substantial modifications are required to make machine-found inference chains easily understandable by humans.

Consequently, the problem of explanation-aware problem-solving amounts to casting the inference path that leads to the solution on a level of granularity that conforms to some sort of basic user communication level. This level is oriented on the pieces of knowledge that make up human domain expertise, which is formulated in terms of rules and domain principles in books and other teaching material. In formal domain, such as mathematics, this level of granularity corresponds to references to axioms, which is called *Assertion Level* [22] in the field of automated theorem proving. Depending on the calculus used for problem-solving, the task to enable adequate explanations requires lifting the representation of the calculus on the user communication level. This may be associated with difficulties and may even be incomplete, especially if the problem-solving component integrates heterogeneous reasoning processes. Examples include references to quantitative methods [27], and the incorporation of results produced by computer algebra systems in deductive reasoners [28]. These partial results are treated as results of external reasoners, which are integrated as a single assertion level step in the solution path representation, the reference to the external reasoner serving as a justification, which is usually a reference to a domain axiom. In order to support flexibility in the presentation, reference to other levels of granularity of the solution path should be provided. This can be done explicitly, through annotations for more abstract forms, such as the application of methods or lemmas which comprise several subsequent assertion level steps, or this can be done on demand, through a systematic transformation process, such as breaking up an assertion level step into a sequence of *Natural Deduction* calculus steps, or by a recomputation performed by an external reasoner, thereby including the exposition of some local trace.

The proper explanation amounts then to a presentation of the solution path, at the basic user communication level, or it can be mixed up with representations on more abstract or occasionally more detailed levels, if such variations are available, and there is evidence on contextual or user-oriented grounds that these alternations are considered superior. In addition, the presentation material should be presented selectively by leaving out contextually inferable reasoning steps to avoid redundancy, which is an extremely important measure in this kind of inference-rich discourse. Another important measure is aggregation of similar assertions, which may appear in some solution descriptions. In natural language presentations, performing aggregation adequately is associated with difficulties, due to subtleties of coordination phenomena (e.g., ambiguities may be introduced). Hence, it should be easier to present a set of expressions underlying a regular pattern in a propositional form, such as mathematical formulas with quantifiers or value ranges.

3.2 Explanations Aiming at Problem Space Understanding

In order to understand why a problem state qualifies as a solution, it is frequently not sufficient to be informed about details of the solution per se. Depending on the nature of the problem, it might be useful to explore alternatives that come close to the solution proposed in terms of some underlying metric. In design tasks, for example, it might be insightful to learn why some solution attempt that superficially looks promising ultimately turned out to be inferior. Such an examination is likely to strengthen the belief in the solution proposed, by getting acquainted with arguments in favor of it, but also with some information about critical aspects, where competitive alternatives exist that could be considered superior with minor goal or assessment modifications.

Unfortunately, examinations of this kind typically require considerable specification effort on behalf of the user and they are associated with a good deal of extra computation to be carried out by the system. The reason for that lies in the organization of problem-solving algorithms, which aim at establishing a solution effectively, thereby cheaply refuting alternations in search paths, so that insufficient documentation for explanation purposes is yielded. A classical example are chess programs, which todate can outperform even the best players, but their explanation capabilities are poor in comparison to their performance. For instance, non-main-variation moves that look worth to be explored for a chess player frequently get assigned a cut-off value in the search tree, which contains no extra information. In order to learn about why such a move does not lead to success, a user has to try out this move, and sometimes several of them in sequence, and to re-start the search engine. Such an exploration is tedious, and it stands in contrast to good human analyses, where critical variations are given, enhanced by functional justifications for moves or sets of moves.

The situation is better for problems which only require shallow search spaces so that detailed information on which results are based can be kept for subsequent reasoning processes. A nice example is the meanwhile historic backgammon program BKG [7], whose strength is mainly based on an elaborate evaluation function [6]. BKG comes with an explanation facility that aims at comparing decisions between two moves which is not done by a detailed and cumbersome listing of all factors involved; instead, it performs an analysis to identify decisive factors in the evaluation scores that *dominates* the other factors [8]. In some earlier work, we have pursued a similar goal for explaining preferences among decisions made by a constraint system, the application domain being room assignment in offices [15]. The idea is to find subsets of constraints that are responsible for the superiority of a design choice over another one, in the sense that they must be relaxed in order to change the preference among the competing design choices, whereas relaxing other constraints would not have this effect. Unfortunately, establishing these dominances requires the exploration of large portions of the overall search space, which is quite in constrast to modern constraint propagation techniques that are well applicable to considerably larger problem spaces.

Similarly, deviations from the inference chain (more generally, inference graph) that constitutes the justification of a solution in proof problems can be examined, which is mostly done for tutoring purposes. Apart from incorrect inferences, which constitute a substantial share of problem step attempts made by students in tutorial sessions, it might be worth to examine correct statements, which may constitute reasonably looking proof attempts that ultimately turn out to be unsuccessful, or they may even contribute to an inference chain justifying the solution, but in a different way than the solution path found by the problem-solving component. Tutoring systems have their problems with solution attempts or requests of this form. For instance, the system ATLAS [32], which tutors students in the area of naive physics, sometimes is incapable to refute a hypothesis made by a student, if the required algebraic computations get too complex. To encounter this problem, tutoring systems frequently precompile solution presentations and partial justifications, associated with appropriate natural language forms, so that they can be activated in dependency of given dialog situations. Thus, explanation is largely decoupled from problem-solving, which holds for the several tutoring systems. Consequently, providing contributions to explanations aiming at problem space understanding appears to be quite hard; it requires recomputations and sometimes selectivity in result presentations.

3.3 Explanations Aiming at the Development of Performance Capabilities

Explanations in this category attempt to convey problem-solving knowledge to the user that is relevant for the issue at stake. A variety of pieces of knowledge and skill-developing advices may serve this purpose, depending on how the underlying task can be addressed. This may range from

1) pure recipes in case a concrete procedure for solving the problem at hand exists, over
2) essentials in searching for a solution – in domains where intelligent tutoring systems apply model-tracing approaches (e.g., basic areas of mathematics and physics), to
3) more abstract and vague hints, typically in domains which cannot be addressed by model-tracing approaches (so-called ill-structured domains, such as law and domain modeling, for instance, building SQL-expressions or Entity-Relationship models).

In case there exists a well-defined procedure for carrying out the required task, explanations on a generic level aim at putting the user in a position to apply this recipe and to address the same problem or sufficiently similar ones on his own. In addition, knowledge about the methods applied may allow the user to assess the reliability of these methods and their potential limitations by checking the information sources and assumptions underlying the task considered.

For domains in which problem-solving is reasonably well understood but requires some heuristic component in concrete operationalizations (thus, where model-tracing is applicable), a variety of techniques have been developed in the field of intelligent tutoring systems which essentially mimic the behavior of human tutors in comparable situations. System reactions vary in dependency of the problem solving context, which comprises the degree to which the problem at hand is solved at some stage, as well as the most recent contributions of the student, which may range from successfully specified inference steps over partially useful specifications to completely useless or erroneous statements. In comparably simple domain, such as high-school algebra, explicit strategies such as partitioning the problem at hand, computing an example, and subsequently abstracting that example into the original problem have been carefully elaborated on the basis of transcripts from human tutoring sessions [14]. They guide the student through the problem-solving process in a few steps, with considerable learning success. For significantly more complex domains, such as naive physics, it is much harder to implement similar strategies as this has been done for high school algebra. A situation in a tutoring session can be addressed from a variety of perspectives, since the objectives for relevant inference steps comprise picking relevant pieces of domain knowledge, their adaquate interpretation and application, and some associated computations. Current systems apply dialog skripts [13] or sequences of hints or help messages in increasing precision [32], which in some sense constitute precompiled tutoring strategies, associated with descriptions of tutoring situations.

For domains where model-tracing is not meaningfully applicable, the problem-solving situation can only be checked in some sort of functional terms. Consequently, not individual contributions but the entire solution is subject to an examination, followed by advice generation. In constraint-based tutors, solutions proposed are checked as to what extent they meet the constraints imposed by the problem specification, so that violated constraints are identified [26]. Feedback about a solution proposed by a student addresses constraints according to their relative importance for the problem at hand, presentations being tailored to the context and expertise of the student.

For explanations in this category, the techniques developed for intelligent tutoring systems give evidence that it is much harder to factor out explanation-aware problem-solving and proper presentation components, as we have attempted to do for the other categories. However, it seems to be more realistic to lower the goal from teaching problem-solving skills to novices to conveying problem-solving techniques or information relevent for this purpose to reasonably knowledgeable users. For doing this, it looks promising to represent more abstract forms of knowledge, such as recipes, descriptions of problem-solving techniques, and meta-information, such as assumptions and sources of knowledge, and tailor and convey them according to situational user needs.

4 Explanation in Tutoring – A Case Study

In this section, we describe the explanation categories outlined in the previous section from the perspective of a tutoring system that teaches theorem proving in some subdomain of mathematics.

4.1 Our Tutoring Environment

The tutoring system referred to in this paper concerns system components and developments in the *Dialog* project. The goal of the project is (i) to empirically investigate the use of flexible natural language dialog in tutoring mathematics, and (ii) to develop a prototype tutoring system that incorporates the empirical findings.

The experimental system engages a student in a dialog in written natural language to help him/her understand and construct mathematical proofs. A modular system architecture is adopted, by use of the proof system ΩMEGA [28]. For tutorial purposes, ΩMEGA has been adapted to handle proofs represented in a human-adequate form [33]. Interaction with ΩMEGA is mediated by a *Proof Manager* [3]. The task of the *Proof Manager* is to communicate with the proof system to check consistency and validity of (possibly ambiguous) interpretations of student utterances within the proof context, and to build and maintain a representation of the constructed proof. Moreover, based on feedback from ΩMEGA, the *Proof Manager* evaluates proof-relevant parts of the utterances with respect to completeness, correctness, and relevance, where *correct* proof steps may not necessarily be *relevant* in that they do not contribute to the progress in finding the proof solution. This categorization is an integral part of our tutorial strategies [31].

To investigate phenomena characterizing written computer-mediated tutorial dialogs, we have collected two corpora of tutor-student dialogs in two Wizard-Of-Oz experiments in the domains of naive set theory and mathematical relations, respectively. The subjects tried to solve three problems in naive set theory (e.g., If $K(B) \supseteq A$, then $K(A) \supseteq B$) resp. mathematical relations (e.g., $(R \cup S) \circ T = (T^{-1} \circ S^{-1})^{-1} \cup (T^{-1} \circ R^{-1})^{-1}$). In the first experiment, the subjects were divided into three groups and tutored with *minimal feedback* (only correctness and completeness evaluation), *didactic* (the expected realization of the given proof-step was included in tutor's feedback), or *socratic* strategy (the tutor guided the student at the solution by using hints, following a pre-defined hinting algorithm), respectively [2]. In the second experiment, the manipulated factor was the presentation of the study-material: verbose (using a mixture of natural language and formulas) versus formal (using mainly formulas). Further details of the setup can be found in [5]. [36] presents the results of the study on the influence of the presentation format.

As the experiments have shown, description in formal domains such as mathematics are characterized by a mixture of natural language and formal expressions. Communicating in this mixed language form turns out to be quite effective, combining the preciseness of formal expressions with occasional sloppiness in natural language descriptions. In order to handle such descriptions, *Dialog* performs mathematical expression processing as part of the parsing component in an architecture for processing informal mathematical discourse [35]. This analysis consists of three stages:

(1) *detection*: mathematical expressions are identified within word-tokenized text;
(2) *syntactic verification:* the identified sequence is verified as to syntactic validity and, in case of a parentheses mismatch, a correction procedure is invoked,
(3) *parsing:* a tree representation of the formula is built.

The parsing component is part of a natural language interpretation component that interfaces the proof manager. Altogether, the basic processing pipeline involves four processing stages, where the three steps listed above form the first stage in the overall procedure: (i) mathematical expression identification and parsing, (ii) syntactic and semantic sentence parsing, (iii) step-wise domain interpretation, (iv) formal representation. As a combined natural language and formula presentation subsystem, we used the system developed earlier for proof presentation purposes [12].

4.2 Explanations Aiming at Providing Justifications

In the context of *Dialog*, this category of explanations concerns presentations of the proof of the theorem at hand or selected portions of it. This breaks down into transformations of the inference graph that represents the proof in the form of the calculus used for proving (typically resolution calculus) to an assertion level proof (the explanation-aware part), followed by various tailoring measures that aim at a presentation in natural language that mimics textbook proofs (the proper explanation part). Details of this process have been presented at the previous meeting of this conference series [18], major original contributions being [16] and [17]. An extension of the techniques originally developed for proof presentation to meet also tutoring purposes lies in the presentation of partial proofs, prominently individual proof steps [10], in accordance with [12].

For adequate use of these procedures in a tutoring session, the approach of using general problem-solving techniques followed by transformations into human-oriented representations turned out to be problematic for a number of reasons:

(1) *Proof results*
When left with methods optimized for problem-solving, which is what theorem provers are made for, the advantage of being quite powerful in finding proofs is to a certain extent compensated by the sometimes unintuitive way in which proofs may occasionally be carried out. The consequence is, that transformations to the assertion level maintain this proof path, which may look quite unnatural to humans. For example, case distinctions are a conceptually meaningful and frequently used method for humans to address a proving task, but this technique is sparsely used by resolution-based provers, so that problems where case distinctions yield a simple and easily perceivable proof path may be solved quite differently by a machine. For example, when proving $|a|\,|b| = |a\,b|$, which easily be done by distinguishing according to positive and negative values of a and b, several automated theorem provers did not apply case distinctions at all.

(2) *Incremental proof development*
In a tutoring session, proofs are not treated as an integrated object that comes as a whole and is to be inspected according to some perspective of interest. A proof is rather developed incrementally, in accordance with contributions of the student and operations done by the tutor, which reflect the state of the dialog within a tutoring session. Hence, the corresponding formal proof that is constructed in accordance with this dialog, must conform to the level of human-oriented inferences in which the proof is developed at all stages of incremental development.

(3) *Proof step checking*
The same considerations as for incemental proof development also hold for the purpose of verifying proof steps proposed by the student: the prover must be able to accommodate the level of granularity addressed, and it must do this for steps of varying complexity; student statements may refer to trivial inference steps, but also to quite complex ones, which require subproofs to be explored in order to make them understandable and verifiable.

Because of these requirements, we have abandoned the approach of transforming machine-found proofs to the assertion level in tutoring contexts. Instead, proofs are carried out on the assertion level itself, in a style that is in accordance with methods developed for proof planning [25], where inferences on a higher level of granularity are made, following recipe-like specifications of domain-specific mathematical problem-solving techniques. [1] shows how reasoning can be based directly on the application of axioms in the mathematical subtheory which the tutoring task is about. Thereby, it is taken care that the problem-relevant inferences are applied and combined in a flexible manner, so that variations in the order of inferences and alternatives in inference steps and even subproofs that lead to the same intermediate results are maintained. The resulting proof structure is then more a graph representing a set of similar proofs with shared parts rather than a single one. In addition, a limited degree of variation in granularity is supported insofar as a subproof that counts as a lemma can be replaced by a reference to this lemma.

4.3 Explanations Aiming at Problem Space Understanding

In the context of *Dialog*, this category of explanations concerns statements about problem solving steps addressed in the tutorial dialog. More specifically, inference steps described by the student are assessed as to their potential contributions to the problem solving process under development. Essentially, this task amounts to tentatively building extensions to a partially developed proof graph, on similar lines as described in the previous subsection. A particular feature thereby is the interpretation of inference step descriptions. From the perspective of the system, this analysis is complicated by various kinds of errors that students occasionally commit, by imprecision and vagueness in the specifications made, and by the sometimes unclear role of these specifications with respect to progress towards solving the problem at hand. In order to address these interpretation problems, *Dialog* has some dedicated analysis components:

(1) *An error-tolerant formula analyzer*
A significant portion of proof step specifications is made in terms of mathematical formulas, mostly embedded in natural language descriptions. As exemplified by our corpora, students produce flawed expressions in many cases, most of them can be traced back to some conceptual confusion, such as using too strong or too weak operators (e.g., equal instead of less or equal resp. less or equal instead of less), or confusing operators with a converse one (e.g., set intersection and union). Based on these observations, formula interpretation comes with a generate-and-test cycle that tries to remedy errors encountered with a limited number of conceptually motivated attempts [20]. Ultimately, a number of hypotheses associated with degrees of plausibility according to the proof context and the modifications made is produced.

(2) *A semantic interpreter that can handle terminological vagueness*
For structurally simple mathematical objects, inference steps can usually be specified in natural language, as an alternative to mathematical formulas. In this form of specification, errors as mentioned above appear in terms of terminological inaccuracies and vagueness, although some of these expressions are established language uses according to textbooks. For instance, "both sets together" can be interpreted as the union or as the intersection of the two sets. Similarly, the expression "*A* is contained in *B*" can be interpreted as a membership or as a subset relation, which can be resolved on the basis of the types of *A* and *B*. These expressions are handled by a two-stage analysis, in which formal expressions are built via explicit intermediate representations that maintain conceptual vagueness [19]. The resulting interpretation typically amounts to a set of alternatives, even in cases resolution is possible, since the tutorial situation may be such that enforcing the student to use exact terminology is considered a tutorial goal.

(3) *A proof step checker that can accommodate partial specifications*
Based on interpreting a student's specifications as outlined in the two preceeding paragraphs, the resulting formal expressions must be compared to the inferential state of the partially developed proof. Interpretation in this context is made difficult by the fact that the formal specifications obtained may be partial or ambiguous. In particular, it may be unclear whether an inference is forward- or backward-oriented, that is, reasoning from the given facts to the goal, or in reverse direction. In order to enable interpretation under these circumstances, underspecification is accepted [1]. The alternative interpretations are tested as to their validity, which, as a by-product leads to the completion of originally incomplete specifications. This way, it can be found out whether or not there is a contextually meaningful interpretation for the student's specifications, including which of the possible interpretations are correct and which ones are not. This information is very valuable for determining the next dialog contribution of the system, in dependency of the tutoring strategy and the context given by the tutorial session.

Altogether, a lesson learned from work on *Dialog* is that the interpretation of explanatory requests, which is not given much emphasis in the literature, may be crucial, although tutoring systems appear to be a special case.

4.4 Explanations Aiming at the Development of Performance Capabilities

In the context of *Dialog*, this category of explanations concerns presentations of feedback about how a proposed contribution to a proof that is interactively developed fits into the state of this problem-solving process. In order to make this assessment, this contribution must be categorized according to its correctness and relevance, which has a strong influence on reactions of the tutoring system. For formally correct proof steps, we distinguish the following degrees of appropriateness:

(1) *Correctness*
A proof step is *merely correct* if it is has no syntactic or logical errors; it may, however, be underspecified in a number of ways, but it is still considered correct if it can be interpreted uniquely within the context of domain axioms. Moreover, the proof step considered fails to meet the criterion of being relevant for the proof at hand (see also the next item in this list).

(2) *Relevance*
A correct proof step is also considered *relevant* if it contributes to making progree towards the proof goal. We consider a proof step as being relevant, if it can be identified as a proof step in at least one of the possible proof variations that have been developed for the proof at hand. This is a strong assumption, since the verification relies on the availability of a set of proofs with sufficient variations in the applied proof methods, covering all reasonable proof paths (a kind of closed world assumption). Such a compromise is unavoidable, given the present insights about proof techniques (see also the discussion about the concept of relevance in [1]).

(3) *Adequacy*
A relevant proof step is also considered *adequate*, if it is not only relevant for the proof at hand, but it is also specified on a level of granularity that is considered appropriate for tutorial purposes. In general, this amounts to a single assertion level step. The justification for this requirement is that the student should be enforced to express a proof in an explicit way, not leaving out inference steps in proof specifications other than elementary ones, such as applications of commutativity. Our experiments have shown that students occasionally make too coarse-grained specifications, which in some cases results from guessing and insufficient justification of inferences. Formal examinations of estimating an appropriate step size on the basis of the proof calculus have been presented in [29].

In addition to these categories, a proof step as specified by the student may contain formal errors, but they may be considered tolerable from a tutorial perspective, in case a (human) tutor can interpret the flawed proof step specification in a meaningful manner. The error analysis and correction component tries to approach these human interpretation capabilities, as well as the reactions observed [21]. If the error correction is considered successful, the system feedback does not only take into account the error observed or the degree of adequacy in which a proof step is specified, but both these factors play a certain role. According to our experiments, reactions by human tutors include mere refusals and various kinds of hints that focus on problem-solving ingredients relevant for the inference step just considered, thus addressing the error only. Occasionally, the focus is put on acceptance of specifications with minor faults by making an explicit correction ("apparently, you meant $\in$ instead of (the last) $\vee$", when the tutor faced the following flawed formula, $\exists z \in M : (x, y) \in R \circ T \vee (x, y) \vee S \circ T$, where the student misued the cut-and-paste facility), or on making an assessment that indicates a mistake in the specifications in such a way that this practically comes out to an acceptance, provided the faulty element is corrected ("that would mean the union instead of the intersection", if the student has used the $\cup$-operator instead of the $\cap$-operator).

The purpose behind hints and other feedback in a tutoring session lies in encouraging the student to carry on with the problem-solving process, by giving away as little information as possible, which is known to increase the learning effect. In tutoring of formal domains, hints and other situational help messages are given, since no generic recipes for addressing a problem are known in most tutoring domains (such as mathematics and most of its subareas).

5 Discussion and Conclusion

In this paper, we have advocated in favor of a high-level categorization of explanations, based on degrees of depth of understanding that explanations in each category aim at. These categories are associated with representation and processing techniques that partition explanations into a an explanation-aware problem-solving part and a proper explanation, that is, presentation part. These techniques differ substantially across the three categories defined in this paper:

(1) *Explanations aiming at providing justification*
(2) *Explanations aiming at problem space understanding*
(3) Explanations aiming at problem-solving skills

For addressing explanations belonging to the first category, partitioning the explanation process into an explanation-aware problem-solving part and a proper explanation presentation part can be organized in a comparably simple manner. The problem-solving process can perform its computations in a way that is widely independent of giving explanations, to allow for efficient searching. At the end, however, the results must be transformed into a form that corresponds to some sort of a "human calculus" (such as the assertion level), yielding a kind of "main variation" justifying the solution, so that these results can be communicated appropriately. In addition to this basic level of granularity, higher levels of abstraction may be of interest, if available. Techniques to set up a reasonable transformation process largely depend on the representation underlying the reasoning process: for logical systems, appropriate definitions of rules underlying a "human calculus" enable suitable transformation techniques. Symbolic mathematical systems (computer algebra systems) may only allow a functional view on the mere results of a computation, especially when the computation is based on some iterative process. For hierarchically decomposable computations, intermediate results can be made available, which may be even at several levels of abstraction, including some kind of summarization of computing steps and even decompositions into finer-grained levels. In contrast to that, constraint systems make their computations in a fundamentally different way, aiming at performance, so that the concept of a "main variation" cannot be identified in their solution space.

For addressing explanations belonging to the second category, it is advisable to perform the problem-solving computations on a level comparable to the "human calculus", as we did in the *Dialog* project, even though there exist more efficient calculi for theorem proving. The main motivation for adapting representations for the reasoning procedure to the level needed for explanation lies in the support for incrementality. The representations of reasoning paths built during problem-solving can be reused to accommodate the reasoning about a specific request. Otherwise, computations would have to be started from scratch for each request that features some alternation in the solution path or in the description of a feasible solution state.

Addressing explanations belonging to the third category essentially comprises three kinds of quite divergent presentations, distinguished according to the degrees of adaptation required:

(1) *Meta-information* – this comprises various kinds of knowledge that needs to be properly represented by indicating the role it bears so that it can be accessed when needed
(2) *Generic recipes* – this piece of information may simply be accessed, but a reasonable use might also be a selective and partially instantiated version accoding to contextual demands
(3) Hints – this kind of presentation is specific to tutoring applications, and it typically comprises variations of pieces of problem-solving knowledge at varying degrees of concreteness

The main purpose of this categorization lies in envisioning a separation of explanation-aware problem-solving form proper explanation presentations, to make the results of problem-solving processes reasonably accessible and to avoid or at least reduce recomputations. Some conditions under which this aim is supported have been discussed and exemplified by a variety of systems.

References

1. Serge Autexier, Christoph Benzmüller, Armin Fiedler, Helmut Horacek, and Bao Vo. Assertion-level Proof Representation with Under-Specification. *Electronic Notes in Theoretical Computer Science* 93:5-23, 2004.
2. Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayovà, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Vo, and Magdalena Wolska. A Wizard-of-Oz Experiment for Tutorial Dialogues in Mathematics. In Supplemantary Proceedings of the *11th International Conference on Artificial Intelligence in Education (AIED-03)*, Vol VIII Workshop on Advanced Technologies for Mathematics Education, 471-481, Sydney, Australia, 2003.
3. Christoph Benzmüller and Bao Vo. Mathematical Domain Reasoning Tasks in Tutorial Natural Language Dialog on Proofs. In Proceedings of the *Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 516-522, Pittsburgh, Pennsylvania, 2005.
4. Christoph Benzmüller, Helmut Horacek, Ivana Kruijff-Korbayova, Manfred Pinkal, Jörg Siekmann, and Magdalena Wolska. Natural Language Dialog with a Tutor System for Mathematical Proofs. Cognitive Systems, LNAI vol. 4429, 1-14, Shanghai, China, 2007.
5. Christoph Benzmüller, Helmut Horacek, Henri Lesourd, Ivana Kruijff-Korbayova, Marvin Schiller, and Magdalena Wolska, DiaWOz-II - A Tool for Wizard-of-Oz Experiments in Mathematics. *KI 2006: Advances in Artificial Intelligence: 29th Annual German Conference on AI*, LNAI vol. 4314, 159-173, Bremen, Germany, 2006.
6. Hans Berliner, On the Construction of Evaluation Functions for Large Domains. In Proceedings of the *6th International Joint Conference on Artificial Intelligence*, 53-55, Tokyo, Japan, 1979.
7. Hans Berliner, Backgammon Computer Program Beats World Champion, *Artificial Intelligence 14*:205-220, 1980.
8. Hans Berliner, David Ackley. The QBKG System: Generating Explanations from a Non-Discrete Knowledge Representation. In Proceedings of the *Second National Conference on Artificial Intelligence (AAAI-82)*, 213-216, Pittsburgh, Pennsylvania, 1982.
9. CALO, 2007. http://www.ai.sri.com/project/CALO.
10. Dominik Dietrich and Mark Buckley. Verification of Proof Steps for Tutoring Mathematical Proofs. In Proceedings of the *13th International Conference on Artificial Intelligence in Education (AIED-07)*, IOS Press, 560-562, Marina del Rey, California, 2007.
11. Francisco Elizalde Flores, Enrique Sucar, Alberto Reyes, Pablo de Buen. An MDP Approach for Explanation Generation. In Proceedings of the *ExaCt 2007 Workshop on Explanation-aware Computing at AAAI 2007*, 28-33, Vancouver, Canada, 2007.
12. Armin Fiedler. Assertion Application in Theorem Proving and Proof Planning. In Georg Gottlob and Toby Walsh (eds.), Proceedings of the *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1343-1344, Stockholm, Sweden, 1999.
13. Art Graesser, Natalie Person, Derek Harter, and The Tutoring Research Group. Teaching Tactics and Dialog in Autotutor. *International Journal of Artificial Intelligence in Education* 12:257-279, 2001.
14. Neil Heffernan and Ken Koedinger. An Intelligent Tutoring System Incorporating a Model of an Experienced Human Tutor. In Proceedings of the *6th International Conference on Intelligent Tutoring Systems (ITS-02)*, 596-608, Biarritz, France, 2002.
15. Helmut Horacek, Explanations for Constraint Systems. In Bernd Neumann (ed.), Proceedings of the *10th European Conference of Artificial Intelligence (ECAI-92)*, 500-504, Vienna, Austria, 1992.
16. Helmut Horacek. Generating Inference-Rich Discourse Through Revisions of RST Trees. In Proceedings of the *Fifteenth National Conference on Artificial Intelligence* (*AAAI-98)*, 814-820, 1998.
17. Helmut Horacek. Presenting Proofs in a Human-Oriented Way. In Harald Ganzinger (ed.), Proceedings of the *16th International Conference on Automated Deduction* (*CADE-16)*, LNAI 1632, Springer, 142-156, 1999.
18. Helmut Horacek. How to Build Explanations of Machine-Found Proofs – A Methodology and Requirements on Domain Representations. In Proceedings of the *ExaCt 2007 Workshop on Explanation-aware Computing at AAAI 2007*, 34-41, Vancouver, Canada, 2007.
19. Helmut Horacek and Magdalena Wolska. Interpreting Semi-Formal Utterances in Dialogs about Mathematical Proofs. In Proceedings of the *9th International Conference on Application of Natural Language to Inforamtion Sytsems (NLDB-04),* Springer, LNCS 3136, 26-38, Salford, UK, 2004.
20. Helmut Horacek and Magdalena Wolska. Handling Errors in Mathematical Formulas. In Proceedings

of the *8th International Conference on Intelligent Tutoring Systems (ITS-06),* Springer, LNCS 4053, 339-348, Jongli, Taiwan, 2006.

21. Helmut Horacek and Magdalena Wolska. Generating Responses to Formally Flawed Problem-Solving Statements. In Proceedings of the *13th International Conference on Artificial Intelligence in Education (AIED-07)*, IOS Press, 17-24, Marina del Rey, California, 2007.
22. Xiaorong Huang. Reconstructing Proofs at the Assertional Level. In Proceedings of the *12th International Conference on Automated Deduction (CADE-94)*, 738-752, Nancy, France, 1994.
23. Deborah L. McGuinness, Alyssa Glass, Michael Wolverton, and Paulo Pinheiro da Silva. A Categorization of Explanation Questions for Task Processing Systems. In Proceedings of the *ExaCt 2007 Workshop on Explanation-aware Computing at AAAI 2007*, 42-48, Vancouver, Canada, 2007.
24. Deborah L. McGuinness, Li Ding, Paulo Pinheiro da Silva, Cynthia Chang. A Modular Explanation Interlingua. In Proceedings of the *ExaCt 2007 Workshop on Explanation-aware Computing at AAAI 2007*, 49-55, Vancouver, Canada, 2007.
25. Erica Melis and Jörg Siekmann. Knowledge-Based Proof Planning. *Artificial Intelligence Journal* 115(1):65-105, 1999.
26. Antonija Mitrovic, Stellan Ohlsson, Brent Martin. Problem-Solving Support in Constraint-Based Tutors. *Technology, Instruction, Cognition, and Learning*. Special Issue on Highlights on AERA 2005, 4(1-2):43-50, 2006.
27. Bruce Porter. A New Class of Knowledge Systems and their Explanation Requirements. Invited talk at the *ExaCt 2007 Workshop on Explanation-aware Computing at AAAI 2007*, Vancouver, Canada, 2007.
28. Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer. Proof Development with OMEGA. In Andrei Voronkov (ed.), Proceedings of the *19th International Conference on Automated Deduction (CADE)*, LNAI 2392, 144-149, Copenhagen, Denmark, 2002.
29. Marvin Schiller, Christoph Benzmüller, and Ann van de Veire. Judging Granularity for Automated mathematics Teaching. In Proceedings of *13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, Pnom Penh, Cambodia, 2006.
30. Peter Spieker, Natürlichsprachliche Erklärungen in technischen Expertensystemen, Dissertation, University of Kaiserslautern, 1991.
31. Dimitra Tsovaltzi, Helmut Horacek and Armin Fiedler. Building Hint Specifications in an NL Tutorial System for Mathematics. In Proceedings of the *17th Florida Artificial Intelligence Research Society Conference (FLAIRS-04)*, AAAI Press, 929-934, Menlo Park, CA, 2004.
32. Kurt VanLehn, Collin Lynch, Kay Schulze, Joel Shapiro, and Robert Shelby. The Andes Tutoring System: Five Years of Evaluations. In Proceedings of the *12th International Conference on Artificial Intelligence in Education (AIED-05)*, IOS Press, 678-685, Amsterdam, The Netherlands, 2005.
33. Bao Vo, Christoph Benzmüller, and Serge Autexier. Assertion Application in Theorem Proving and Proof Planning. In Georg Gottlob and Toby Walsh (eds.), Proceedings of the *18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 1343-1344, Acapulco, Mexico, 2003.
34. Michael Wick and William Thompson. Reconstructive Expert System Explanation. *Artificial Intelligence* 54(1-2):33-70, 1992.
35. Magdalena Wolska and Ivana Kruijff-Korbayovà. Analysis of Mixed Natural and Symbolic Language Input in Mathematical Dialogs. In Proceedings of the *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 24-32, Barcelona, Spain, 2004.
36. Magdalena Wolska and Ivana Kruijff-Korbayovà. Factors Influencing Input Styles in Tutoring Systems: The Case of the Study-Material Presentation Format. In Proceedings of the *ECAI-06 Workshop on Language-Enabled Educational Technology*, 86-91, Riva del Garda, Italy, 2006.