Explanations in Cooperative Problem Solving Systems

Thomas Mastaglio and Brent Reeves

Department of Computer Science and
Institute of Cognitive Science
University of Colorado, Campus Box 430
Boulder, CO 80309
email: brentr@boulder.colorado.edu

Abstract

It is our goal to build cooperative problem solving systems, knowledge-based systems that leverage the asymmetry between the user's and the system's strengths and thus allow the dyad of user and computer system to achieve what neither alone could achieve. Our experience has shown that in these cooperative systems, the need for explanations is even more evident than in traditional expert systems. This is due to the fact that these new systems are more open-ended and flexible and therefore allow for more possibilities in which a user can reach an impasse, a point at which it is not clear how to proceed.

Observation of human-human problem solving shows that people are sensitive to the domain under discussion and the other's knowledge of that domain. People tend to construct explanations that are minimal in the number of concepts or chunks. These explanations are not comprehensive, and the communication partner is able to follow up on aspects which are still unclear.

1 Introduction and Theoretical Basis

Our research focuses on how to build cooperative knowledge-based systems that take advantage of the different strengths of users and computer systems. Computers can be a source of expert domain knowledge, knowledge they can use to make suggestions to users. The computer system's role in this dyad must include the ability to explain those suggestions. For various reasons, current explanation systems often fail to satisfy users. Too frequently they are based on an implicit assumption that explaining is a one-shot affair and that the system will be able to produce or retrieve a complete and satisfying explanation provided it is endowed with *artificial intelligence*.

Our approach takes advantage of existing information and knowledge-based systems technology already available to provide the user access to explanations at different levels of detail and complexity. Development efforts focused on the concepts requiring explanation rather than on selecting a complete prestored explanation. A conceptual model of the domain and a user model provide that set of concepts. Explanation giving requires establishing a suitable conceptual context for a dialogue between the user and the system. The system must know what concepts are required to understand an entity in the domain and furthermore which subset of those concepts are unfamiliar to a given user. Forming such an idiosyncratic explanation is a construction rather than a selection process.

We are investigating how to design systems that serve users as they are actively engaged in their own work — cooperative problem solving systems. These systems provide a task-based environment which users employ to accomplish some goal. We want them to be more than a communication medium with which to describe a problem. In the context of these task-based environments we analyzed the

reasons people seek explanations. The common triggering condition is that they experience an impasse that stops or limits their work. We call these "task-oriented impasses" and have cataloged them to better understand how explanation giving can help to overcome the impasses. The four categories of task-oriented impasses are: action impasses, communication impasses, motivation impasses, and curiosity impasses.

1. *Action impasses* occur when users do not know what to do next. Some action impasse questions are: What should I do next? Is *action* the right thing to do next? How do I do *action*? What did the system just do? What are the results of doing *action*? Can I do *action* now?

2. A *communication impasse* is a failure to understand a given object in the environment. Representative questions are: What is *object*? Why is *object1* shown instead of what I expected, namely *object2*?

3. *Motivation impasses* fall into the realm of behavioral psychology; their basis is an anthropomorphic view of the computer system: Representative questions are: Why did the system do *action*? Why did the system just communicate with me? Why did the system just say *X*? Why should I do *action*? Why is $action_1$ better than $action_2$?

4. *Curiosity impasses* are a bit different. The other categories consist of questions that arise when users encounter a problem. Curiosity impasses are not necessarily impasses, in a strict sense, but rather a diversion from continuing with the task. They are circumstances in which users seek to gather information that is interesting or might be helpful, but *the lack of which* will not preclude them from further work. For consistency we will refer to them also as "impasses". Questions that illustrate curiosity impasses are: Is *object* a concept *X*? How do $object_1$ and $object_2$ differ?

To overcome these impasses, explanations in cooperative problem solving systems serve four functions; functions that are similar to findings of empirical work on explanation giving in expert systems [16]. The functions of explanations in cooperative problem solving systems are:

1. To allow users to examine the system's recommendations,

2. To relate recommendations to domain concepts — understanding "what is suggested",

3. To provide a rationale for recommendations — understanding "why this would be better",

4. To educate a user about underlying domain concepts.

These four functions are not mutually exclusive. A single explanation episode by a cooperative problem solving system will often accomplish several of these functions.

Even in cases where explanation systems have been designed with the above functions in mind, the result has typically been a system that explains too much, and therefore not enough. A familiar example to many computer system users is the *man page*, which contains everything a user might want to know, from the system's perspective, and for that very reason is often too much to be of use.

In observing problem-solving interactions between salesmen and customers in a large hardware store, we noticed that explanation never took the *man page* approach. When explanations were required, the approach was one of minimizing the explanation, then following up on unclear concepts when necessary [13]. This is interesting if you consider the fact that the particular store carries over 350,000 different items in over 33,000 square feet of retail space. If salesmen took the approach found in many computer systems, the explanations they gave would be extraordinarily long, in order to be complete, and complex, in order to take into account the relationships to other items in the store.

As an example of computer systems with extensive documentation, consider the Symbolics Lisp machine, which comes with over 4000 pages of documentation, most of which is available online in hypertext format. Issuing a query (i.e. a request for an explanation) can result in pages and pages of text and examples; frequently too much to digest and make sense of. As computer systems increase in power, they also increase in complexity. Simply having more powerful computers will not solve the problem of explanation, it will only exacerbate it.

In order to model the "minimalist explanation" approach found in human-human interactions in a knowledge-based system, we designed an extension to LISP-CRITIC, a knowledge-based source transformation system. Supporting idiosyncratic explanation giving requires a user model that contains a detailed representation of users' knowledge. The explanation approach needs something more than classifying users by level of expertise. Our framework uses a fine-grained user model and a minimalist approach [6]. More theoretical basis for the minimalist explanation approach is found in related work on discourse comprehension:

1. Short-term memory is a fundamental limiting factor in reading and understanding text [2, 1]. The best explanations are those that contain no more information than absolutely necessary, since extra words increase the chances that essential facts will be lost from memory before the entire explanation is processed.
2. It is important to relate written text to readers' existing knowledge [11, 5].

Similar guidelines are also offered as principles of rhetoric. Flesch developed formulae to evaluate the readability of text [9] that emphasize brevity. Computer explanation systems should comply with similar standards, therefore short sentences and known vocabulary as important criteria. Other support is found in Strunk and White's familiar manual; it contains similar advice. Writers of explanatory text are told "Don't explain too much" [15].

2 An Explanation Approach

Supporting explanation requires the system to have enough knowledge to describe what is going on and why. Operational knowledge is often captured in rule form (If *condition* then *action*). For users to understand a rule means that they must know the concepts underlying a rule. A conceptual domain model provides the entire set of prerequisite knowledge and a user model filters that set for an individual.

The user model helps select which subexplanations to actually present. One view of explanation is that users are engaged in an understanding process that is based on generating explanations to themselves [14]. It follows that the explanation component has to provide users with sufficient information so they can develop their own self-explanation, and yet not with so much information that it interferes with a person's ability to construct meaningful explanations.

Determining what to explain to a user requires decomposition; the system must execute a GPS-like process [3]. Prerequisite knowledge is the knowledge a user must know in order to understand a given concept. This is a recursive process, understanding those domain concepts that are prerequisites for the given concept requires, in turn, understanding their prerequisites and so on. To support such an approach a deep structure for the domain is queried to obtain the set of prerequisites. Still, a satisfactory explanation approach will do more, namely identify the concepts in that set that do not need explaining because the user already knows them. Then the system reasons about the best way to explain the remaining concepts. Based on this general schema, we developed a framework consisting of several

levels of explanations.

It is also important to incorporate a fallback capability, allowing the user some recourse when the initially provided explanation fails. No computer-generated explanation system can be expected to satisfy users completely — people are not able to achieve this and we do not expect any more of computer systems. The situation where a user does not understand an initial explanation or wants more detailed information must be accommodated. One technique is the reactive approach [12]; our approach is simpler in that it makes use of existing hypertext capabilities.

3 LISP-CRITIC Explanation System

We have developed an explanation component for a LISP critiquing system in accordance with this framework. LISP-CRITIC is that system [7, 4]. It is a knowledge-based interactive system that operates in a programmer's editing environment, providing suggestions on how to improve user LISP programs. When called on to examine a piece of code, the system informs the programmer of each suggestion it has for improving the program. As each suggestion is provided, the programmer can accept the suggestion, reject it outright or ask for an explanation. Users seek explanation to help them understand the suggestion well enough to make the accept/reject decision. When a suggestion is accepted the system actually rewrites that piece of code in the user's editing buffer.

We developed an approach that provides information in four layers of increasing detail. The first two layers are not necessarily explanations in a specific sense of the word but we found that in many situations they provide enough information to satisfy system users.

1. A fundamental piece of information is the name of the rule that identifies a suggested transformation. The rule name is an abstract reference to a chunk of domain knowledge, that chunk is an instance of the domain object class LISP-CRITIC rule; it may or may not have meaning to users. Sometimes when it does have meaning, users are satisfied just knowing which rule fired and no further explanations is required. They have sufficient information on which to base an accept/reject decision.

2. The transformed code, as recommended by the system, can be compared to the user's code. The system shows the user's code and its recommended version. Often this is sufficient for users to understand the suggestion and they can make a decision on whether to accept or reject the advice.

3. The minimal explanation layer is the point where our theoretical investigations comes into play. The user is provided with a textual description of the system's advice based on the domain concepts behind that transformation.

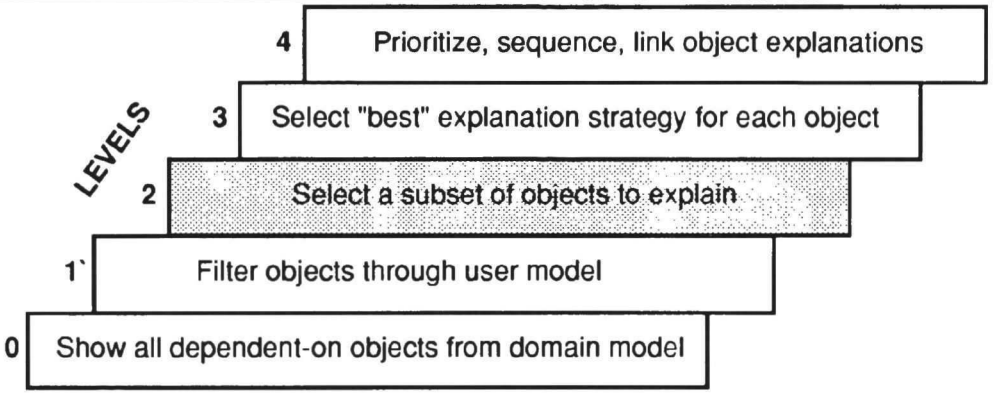
4. The underlying computational environment already includes a hypertext-based information space, the Symbolics Document Examiner. LISP-CRITIC facilitates access to this information for users who are not satisfied with the minimalist explanation of the advice. Users navigate through the hypertext space themselves; the system first locates them in an appropriate context.

The approach used in the current implementation evolved from attempts to provide explanations for an earlier version of LISP-CRITIC [10]. The approach taken involved rule-tracing and prestoring textual descriptions for each transformation (layer 1 and 2 above). A suite of alternative canned explanations for each rule were provided; each one designed to meet the needs of a user with a particular level of expertise. To provide the correct explanation, the system classified a user as a novice, intermediate or expert programmer. This approach was of limited success. A major finding from that work was the need

for a finer grained approach to modelling individual users than classification: An approach that also supports dynamic update as users' expertise changes.

The theoretical framework discussed above led us to emphasize concise and readable explanations. Metrics with which to evaluate, or guidelines to help construct such explanations do not exist. We decided that it would be best to provide terse explanations tailored individuals, while recognizing that at times users require additional information. To provide that information, we provided the hypertext "hooks" into an existing on-line documentation system.

The explanation levels shown in Figure 1 capture the necessary and sufficient conditions for an adequate explanation, each level incrementally improving on the work done at a lower level by using additional knowledge about the user and the domain. A level 0 explanation does not require knowledge about individual users. It uses the conceptual domain model to meet the necessary condition that it knows everything required to understand the entity needing to be explained. The set of prerequisite concepts required in order to understand the object the user wants explained is provided by a deep domain model. Level 1 brings the user model into the process; the prerequisite concepts are "filtered" through the user model to determine the subset of them appropriate for explaining to a given individual. What remains is often larger than what would be reasonably explained in a single dialogue episode. Therefore the explanation component uses knowledge about what makes a good explanation at level 2 to order and prioritize the concepts that are to be explained.



Five levels of explanation have been identified. Level 0 insures that all prerequisite knowledge for a given domain object is available to the explanation component. Level 1 builds on top of level 0 and so forth. The current implementation provides level 2 explanations. The domain and user model can also support levels 3 and 4.

Figure 1: Explanation Levels

Level 3 makes use of additional domain knowledge or perhaps other information in the user model to determine a "best" strategy for explaining each concept. For example, a system could make use of the links between objects in the domain model and the user model to determine candidate concepts or functions for use in a differential description [8]. One object can be described differentially in terms of another object that the user already knows. Level 4 performs "syntactic sugaring". Here the individual explanations from level 3 are ordered and linked in an appropriate manner. This is a nontrivial process

and requires the system to have knowledge of discourse as well as natural language generation capabilities that exceed the present state of the art.

4 Role of the user model in explanations

Cooperative problem solving systems must tailor explanations in order to adequately perform the four functions of explanation. The basis for this tailoring is the user model. The user model is an essential component. A simple approach is to classify users by their expertise (e.g., novice, intermediate, expert). The use of stereotyping and classification schemes can accommodate those systems that provide one-shot advise but cooperative problem solving systems need a finer grained representations of the user.

The user model contains a representation of the user's domain knowledge adequate to support any of the five "levels" of explanation shown in Figure 1. The implication is that it must be based, at least partially, on the conceptual model of the domain in order for it to serve as the filter for level 2 explanations. The model needs to capture the user's goals in order to support level 3 explanations.

Supporting level 4 explanations are more difficult. Explanations at this level are at the frontier of research on natural language generation and human-computer dialogue technology. We will not know everything required of a user model to support explanations at this level until that research matures. At this point we conjecture various user model capabilities needed to support this level, such as knowing the education and reading comprehension level of a user.

5 Summary and Conclusions

The problems with current explanation systems are widely recognized, but most efforts to improve them attempt either to emulate human-to-human communication in inappropriate ways or to provide a complete explanation in "one shot". Theoretical results in discourse comprehension and principles of rhetoric are a suitable starting point but it must be kept in mind that all human-to-human communication techniques are not appropriate for computer-based explaining.

In studying human-human problem solving, we note that one aspect that is less important than expected is the ability to produce and understand natural language. People rarely talk in grammatically correct or even complete sentences, yet they manage to communicate and solve problems. We call this 'natural communication,' rather than natural language. We have attempted to map natural communication over to human-computer systems by supporting minimalist, layered explanations with the capability for further follow-up via hypertext.

The approach we used provides several layers of explanation for advice from a knowledge-based system. The first two layers are not explanations in the strictest sense, although they can help users achieve understanding, but are detailed descriptions of what was recommended. The 3rd layer clarifies the recommendations and exposes the user to the underlying rationale for that recommendation. These are minimal explanations that query a user model to find out what is necessary for the user to understand a LISP concept, LISP function, or LISP-CRITIC rule. The highest layer of information provides users a fallback capability using a rich hypertext information space in they are free to explore details or examine concepts which they still do not understand.

References

[1] Bruce K. Britton, John B. Black (editors).
Understanding Expository Text.
Lawrence Erlbaum Associates, London, 1985.

[2] T.A. van Dijk, W. Kintsch.
Strategies of Discourse Comprehension.
Academic Press, New York, 1983.

[3] G.W. Ernst, A. Newell.
ACM Monograph Series: GPS: A Case Study in Generality and Problem Solving.
Academic Press, London - New York, 1969.

[4] G. Fischer.
A Critic for LISP.
In J. McDermott (editor), *Proceedings of the 10th International Joint Conference on Artificial Intelligence (Milan, Italy)*, pages 177-184. Morgan Kaufmann Publishers, Los Altos, CA, August, 1987.

[5] G. Fischer, S.A. Weyer, W.P. Jones, A.C. Kay, W. Kintsch, R.H. Trigg.
A Critical Assessment of Hypertext Systems.
In *Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.)*, pages 223-227. ACM, New York, May, 1988.

[6] G. Fischer, A.C. Lemke, T. Mastaglio, A. Morch.
Using Critics to Empower Users.
In *Human Factors in Computing Systems, CHI'90 Conference Proceedings (Seattle, WA).* ACM, New York, April, 1990.

[7] G. Fischer, T. Mastaglio.
Computer-Based Critics.
In *Proceedings of the 22nd Annual Hawaii Conference on System Sciences, Vol. III: Decision Support and Knowledge Based Systems Track*, pages 427-436. IEEE Computer Society, January, 1989.

[8] G. Fischer, T. Mastaglio, B.N. Reeves, J. Rieman.
Minimalist Explanations in Knowledge-based Systems.
In *Proceedings of the Twenty-Third Annual Hawaii Conference on System Sciences, Vol. III: Decision Support and Knowledge Based Systems Track*, pages 309-317. IEEE Computer Society, January, 1990.

[9] R. Flesch.
The Art of Readable Writing.
Harper & Brothers, New York, 1949.

[10] J. Frank, P. Lynn T. Mastaglio.
Using A Critic Methodology as a Computer-aided Learning Paradigm: extending the concepts. 1987.
Final Project Report for CS659 - Fall Term 1987.

[11] W. Kintsch.
The Representation of Knowledge and the Use of Knowledge in Discourse Comprehension.
Language Processing in Social Context.
North Holland, Amsterdam, 1989, pages 185-209.
also published as Technical Report No. 152, Institute of Cognitive Science, University of Colorado, Boulder, CO.

[12] J. Moore.
A Reactive Approach to Explanation.
Technical Report, USC/Information Sciences Institute, 1988.

[13] B. Reeves.
Finding and Choosing the Right Object in a Large Hardware Store -- An Empirical Study of Cooperative Problem Solving among Humans.
Technical Report, Department of Computer Science, University of Colorado, Boulder, CO, 1990. forthcoming.

[14] R.G. Schank.
Explanation Patterns: Understanding MEchanically and Creatively.
Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[15] W. Strunk, E.B. White.
The Elements of Style.
Harcourt-Brace?, New York, 1957.

[16] J.W. Wallis, E.H. Shortliffe.
Customized Explanations Using Causal Knowledge.
Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.
Addison-Wesley Publishing Company, Reading, MA, 1984, pages 371-388, Chapter 20".