July 1994

Critiquing: Effective Decision Support in Time-Critical Domains (Dissertation Proposal)

Abigail S. Gertner
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-94-35.

Critiquing: Effective Decision Support in Time-Critical Domains (Dissertation Proposal)

Abstract

The effective communication of information is an important concern in the design of an expert consultation system. Several researchers have chosen to adopt a *critiquing* mode, in which the system evaluates and reacts to a solution proposed by the user rather than presenting its own solution. In this proposal, I present an architecture for a critiquing system that functions in real-time, *during* the process of developing and executing a management plan in time-critical situations. The architecture is able to take account of and reason about multiple, interacting goals and to identify critical errors in the proposed management plan. This architecture is being implemented as part of the TraumAID system for the management of patients with severe injuries.

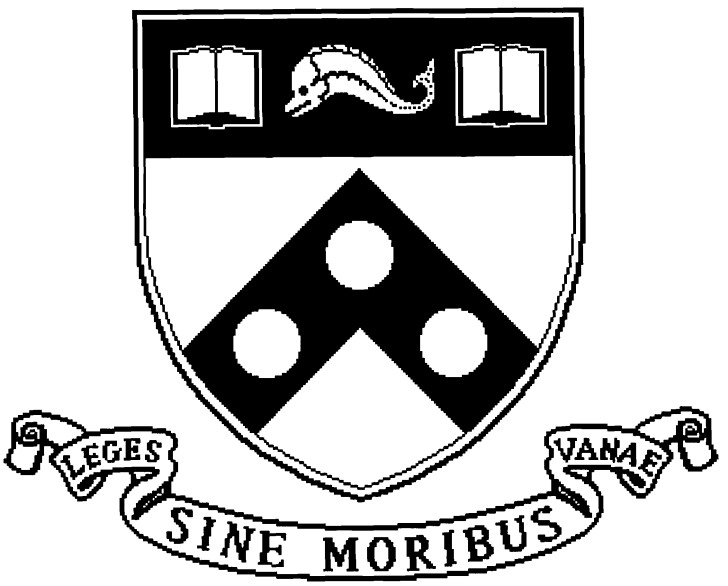
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-94-35.

Critiquing: Effective Decision Support In Time-Critical Domains (Dissertation Proposal)

MS-CIS-94-35
LINC LAB 272

Abigail S. Gertner



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

July 1994

Critiquing: Effective Decision Support in Time-Critical Domains

Dissertation Proposal

Abigail S. Gertner
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
agertner@linc.cis.upenn.edu

Abstract

The effective communication of information is an important concern in the design of an expert consultation system. Several researchers have chosen to adopt a *critiquing* mode, in which the system evaluates and reacts to a solution proposed by the user rather than presenting its own solution. In this proposal, I present an architecture for a critiquing system that functions in real-time, *during* the process of developing and executing a management plan in time-critical situations. The architecture is able to take account of and reason about multiple, interacting goals and to identify critical errors in the proposed management plan. This architecture is being implemented as part of the TraumAID system for the management of patients with severe injuries.

Contents

List of Figures

Chapter 1

Introduction

Imagine a situation in which a relatively inexperienced resident surgeon is treating a patient who has just been brought in to the hospital with a gunshot wound in the abdomen. The patient is in shock and is losing blood rapidly. The resident decides to do a CT-scan of the abdomen to find the source of the bleeding, and then go straight to the operating room. The attending physician watching the procedure interrupts the resident to inform him that an abdominal x-ray would provide the same information as a CT-scan and would be faster. He also mentions that the resident should probably get an x-ray of the chest to make sure that the bullet did not travel upward and cause injuries in the chest cavity.

What does the advisor have to do in order to provide this kind of assistance? He must have an understanding of the resident's beliefs and goals, in order to evaluate and address any misconceptions that might underlie the intention to carry out a less than optimal plan. He also must understand the problem, and the approach he considers best for addressing it, in order to be able to offer alternatives to the faulty plan. In addition, the advisor should have some idea of how to communicate cooperatively in order to influence the resident's future actions.

1.1 Critiquing in TraumAID

The TraumAID system, which has been under development at the University of Pennsylvania for almost ten years, is a tool for assisting physicians during the initial definitive management phase of patients with severe injuries.[1] During this phase, which is often characterized by the need for urgent action, preliminary diagnoses of the patients are pursued and initial treatments are carried out.

The current system, TraumAID 2.0, links a rule-based reasoner, that derives conclusions and goals to pursue from the available evidence about the patient, and a planner, that constructs a (partially ordered) plan for how best to address the

[1] Currently, the system's knowledge base is restricted to penetrating injuries to the chest and abdomen.

currently relevant goals [41, 49]. A more detailed discussion of the system appears in Chapter 4.

In response to any new information regarding the patient or the management procedures carried out so far, TraumAID 2.0 outputs a listing of its current recommended management plan, which can be taken literally as orders for subsequent action by the physician, or more loosely as a guide for deciding on subsequent actions. Viewing TraumAID's output as orders, though, fails to take into consideration the fact that physicians are intelligent agents, capable of reasoning on their own about how to manage their patients. It is unreasonable to expect physicians to abandon their own decision-making skills to follow the recommendations of a computer system. In addition to the psychological drawbacks of letting the system pre-empt a physician's decision making, it is less efficient, since there is actually no need for him to refer to the recommendations of the system unless there is a problem with the way he is managing the patient.

However, if the system's recommended management plan is interpreted merely as a guide for deciding on subsequent actions, the system will have nothing to say when the physician's decisions diverge from its recommendations. Once he decides to go against TraumAID's proposals, the physician is, in effect, on his own. What is needed in this situation is a system that can detect problems with a physician's intended management plan *when they arise* and present its recommendations in terms of explanations and possible alternative courses of action *in the context of the physician's intended actions.* The system should not aim to *replace* the physician's skills, but rather to *augment* them.

1.2 Statement of Thesis

In this proposal, I will present an approach to collaborative problem solving using a *critiquing* interface to simulate this sort of advising behavior in a computer system. The approach I present is appropriate for decision-support in domains that have the following features:

1. **Multiple goals**: Trauma management often involves reasoning about multiple interacting goals. Therefore, the physician's proposed actions must be evaluated globally, in the context of the entire plan. Additionally, the presence of multiple goals with varying degrees of urgency means that the critique must address not only what actions are performed but also the order in which they are carried out.

2. **Time constraints**: In trauma management, diagnosis and treatment of different actions are carried out within the same time frame, and the urgency associated with actions makes it important to intervene within a short period

of time. The critiquing system must therefore be capable of constantly updating its representation of the situation and the user's plan, and revising its critique based on new information.

3. **Task-centered activity**: In an emergency situation such as trauma resuscitation, the physician's primary focus of attention is reserved for the task of caring for the patient. Therefore, the amount of work that he has to do to understand the critique should be kept to a minimum. The system should avoid saying anything that (a) the physician already knows, (b) is "common knowledge" to anyone who would be using the system, or (c) reflects trivial differences between the system's preferences and the physician's.

4. **Gap between order and execution**: Actions that involve resources that need to be brought to the trauma bay or that can only be done elsewhere must be *ordered* prior to being carried out. Since orders can be rescinded, comments pointing out problems with an order can potentially make a clinically significant difference to patient management.

I will argue first that in such domains, human-computer interaction based on a process-oriented propose-and-critique model is preferable to a more prescriptive advising mode. Furthermore, I will claim that the ability to critique effectively depends on a combination of integrated knowledge structures and reasoning capabilities:

1. A plan inference component that uses knowledge about actions and goals in the domain, together with knowledge about the specific situation, to infer and continually update a model of the user's goals and intentions from his proposed actions.

2. A plan evaluation component that makes use of knowledge about causal factors, policy, practice guidelines, etc. and how they should shape behavior in a given situation, in order to identify potentially significant errors that will then be mentioned in the critique.

3. A critique generation component that converts the results of plan evaluation into a concise and coherent natural language critique.

Finally, I will demonstrate that expert critiquing can be simulated in a computer system through the description of a critiquing module that I have implemented as part of the TraumAID system for trauma management.

1.3 Proposal Outline

The next chapter, explains the motivation for interacting using the critiquing approach, and introduces the issues that I believe are important for the design of a

critiquing system. Chapter 3, reviews the expert critiquing literature, relating the different theories and implementations to issues discussed in the previous chapter. Chapter 4 describes my implementation of TraumaTIQ, the critiquing module for TraumAID. Finally, Chapter 5 concludes by restating the main thesis and discussing the work that remains to be done as part of this project.

Chapter 2

Issues for Critiquing System Design

Over the past decade, the term *critiquing* has been applied to a wide range of applications including therapy planning, knowledge base acquisition, computer aided design, software engineering, and desktop publishing (see [44]). What these systems have in common is that they take a problem description and a proposed "solution" or "design" as input from the user and produce some kind of commentary aimed at improving the correctness, efficiency, clarity, and/or workability of the solution. This is in contrast to more traditional expert systems, which simply take a description of the problem and use their domain knowledge to produce a solution. Figure 2.1 illustrates the difference between expert systems and critiquing systems.

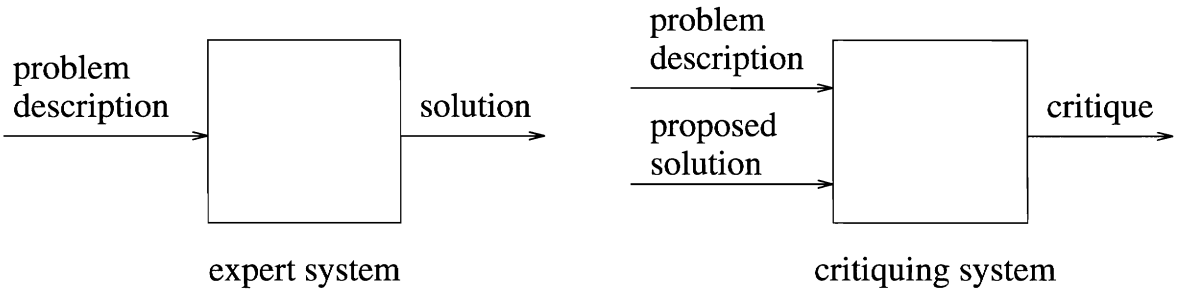


Figure 2.1: The expert system model vs. the critiquing model

In this proposal, I present an approach to critiquing in real-time that is based on a model of cooperative communication between intelligent agents. Each set of proposed actions input by the user is treated as an implicit query: "Is this (partial) plan acceptable in the current situation?" Like other models of cooperative response generation (CRG), I will be concerned with issues of plan inference, plan evaluation, user modelling, and response planning. (For a discussion of the range of work on CRG, see the literature review in [7].) However, unlike other work in this area, which has focused on the interaction between an "expert" system and a "novice"

user, I am modelling the interaction between two experts or near-experts who share a common top-level goal – managing the patient as effectively as possible. The role of the system, therefore, is that of an agent who observes and evaluates the user's behavior in terms of what it considers to be the optimal plan according to a set of general and specific metrics, and responds only if a significant problem is detected.

2.1 Motivations for critiquing

The integration of medical decision-support systems into clinical environments has been a widely recognized problem ever since such systems began to appear. While recognizing their potential for improving the quality of patient care and for controlling costs, physicians have tended to reject new technologies which they see as intrusive, time-consuming, or a challenge to their judgment or autonomy as clinical decision-makers [2]. The approach a system uses for human-computer interaction, therefore, plays an essential role in its ultimate effectiveness.

In general, there are several advantages of using a critiquing approach for decision-support rather than the more standard expert system approach [28]:

- Acceptability: The difference in perceived roles of human and computer can affect the psychological acceptability of the system to its users:
  - A critiquing system can be seen as assisting the user in developing his plan rather than presenting a contrary solution.
  - Critiquing systems can be less intrusive by producing comments only when a significant problem is detected.
  - While expert systems traditionally assume the primary decision-making capacity, treating the user as a passive follower, critiquing systems take a secondary role in decision making, leaving the primary control in the hands of the user.
  - Rather than presenting a solution that may or may not be similar to what the user was thinking of, the critiquing approach provides a user-centered evaluation of the problem.
- Flexibility: Certain domains (such as medicine) in which expert systems have frequently been developed, are characterized by a significant degree of variation in what can be called an "acceptable solution."
  - Practice variability, due to differences in training, expertise, and available resources, means that there is seldom one correct way to approach a problem.
  - Subjective judgments, which cannot easily be modelled as part of an expert system, are often an essential aspect of decision-making.

Critiquing systems can accommodate these kinds of variation by allowing for a range of acceptable solutions.

Given these characteristics, the trauma management domain seems to be a promising candidate for the critiquing approach. A critique can provide a focussed discussion of the physician's proposed actions, serving to remind him of items he may have overlooked while refraining from explicitly telling him what to do. In contrast, a system that presented its recommended plan and required the physician to interpret it in terms of his own reasoning about the case would require a great deal more attention from the physician, and could be rejected as an attempt to replace his judgment with that of a machine.

2.2 The Types of Errors Found in Physicians' Plans

If physicians always developed and executed plans that were in perfect compliance with what would accord with perfect knowledge and perfect judgment, there would be no need for a system like TraumAID. Unfortunately, however, the care given by even experienced trauma surgeons is often sub-optimal, although these problems do not always have an effect on the patient outcome. Support for this claim comes from the analysis of data from a study evaluating the performance of TraumAID 2.0 (see [9, 41]). This analysis suggests that the actual performance of physicians on real cases is not always acceptable to experts in the field of trauma surgery. When expert judges were asked to compare the management plans created by TraumAID 2.0 to the actual care given to patients, they rated the actual care as unacceptable in 14 out of 97 cases, compared to 4 unacceptable ratings for TraumAID 2.0. Some of the most common errors pointed out in the physicians' management were (1) the *overuse of unjustified and risky diagnostic procedures*, (2) *omission of appropriate therapy*, and (3) *failure to perform urgent actions promptly.*

The occurrence of errors in management plans suggests that the physicians responsible for the delivery of care sometimes have incorrect or missing knowledge, which can be counteracted with a simple reminder, or they experience lapses in judgment, such as those described in the literature on heuristic biases in decision-making [17]. From the point of view of critiquing, it may be advantageous to be able to detect when such biases might be influencing a physician's decision-making.

The process by which people make judgments and come to conclusions with incomplete knowledge and in uncertain situations has been the subject of numerous experimental studies, such as those presented in [17]. For example, Kahneman and Tversky have demonstrated that people making judgments based on uncertain information make use of heuristic rules to guide them toward conclusions. These heuristics are derived from everyday experience and, as such, are often useful in

simplifying complicated situations in order to decide what to do. In certain circumstances, however, they can also lead to systematic, predictable biases [47]. These biases have been demonstrated not just in untrained subjects reasoning about an unfamiliar domain, but also in the reasoning of experts, such as surgeons, who are trained in their area of expertise and may also have some background in statistics and formal reasoning [8, 43].

In Chapter 3 I will describe an approach to critiquing that makes explicit use of knowledge about these types of judgment biases [43]. While this is not the approach I take in my work, I do believe that an understanding of typical judgment biases can be incorporated into the procedures the system uses for inferring and evaluating a physician's plans. For example, the *availability* heuristic is often used to judge the frequency of items in a class or the expected likelihood of a particular event. The more easily members of the class can be recalled, or an occurrence of the event can be imagined, the higher the judgment of frequency will be. This rule is reasonable in many cases because, all else being equal, a more frequent item or event will be easier to bring to mind. However, there are several other factors that contribute to the cognitive availability of a class, such as the salience of its members or the complexity of the procedure needed to conceptualize them. The availability heuristic has been shown to bias judgments in favor of the more easily conceptualized classes. In the context of patient management, the availability bias suggests that physicians might tend to omit relevant tests and jump to conclusions about a diagnosis on the basis of insufficient but easily available evidence. Therefore, in trying to understand an unmotivated action, it might make sense to consider whether that action is intended as therapy for a diagnosis that is only partially justified by easily recalled or conceptualized information.

Another heuristic discussed by Kahneman and Tversky is *representativeness*, which is used to judge whether a given instance is a member of a class. The more characteristics of the class an instance has, the more likely it will be judged to belong to that class. Unfortunately, this heuristic fails to take into account a number of factors. The size of the sample can be an important consideration: a small sample is less likely to conform to the general population. The representative heuristic is also insensitive to the base-rate frequency of the outcomes, so that the probability that an instance belongs to a certain class will be judged to be higher if it is highly representative of that class, regardless of how infrequent instances of the class actually are in the general population: a patient who has symptoms consistent with a diagnosis of appendicitis, but whose symptoms are identical to the textbook description of some extremely rare disease is still more likely to have appendicitis purely on the basis of the much higher rate of appendicitis in the general population [8]. Failure to pursue appropriate therapy may therefore be the result of an incorrect diagnosis resulting from a representativeness bias.

2.3 Critiquing in Different Domains

As we will see in Chapter 3, the critiquing approach has been used in a wide variety of applications in many different domains. It is not surprising that the interpretation of how a critiquing system works varies quite a bit from one application to another:

- Some critiquing systems develop their own solution to the problem and some do not.
- Some require a complete solution before they present their critique, while others generate an ongoing critique throughout the development of a solution or plan.
- Some focus on the generation of the actual *text* of the critique, while others are more concerned with the knowledge representation necessary to generate comments.

In general, then, one would like to be able to describe how the characteristics of a system's domain influences its requirements for critiquing. This issue was explored by Perry Miller in a series of prototype critiquing systems [28]. Miller was interested in identifying domain characteristics that would lend themselves to the critiquing approach. He also looked at how different aspects of critiquing might be more or less important in different domains. This experience led him to identify two dimensions along which domains vary. Where a domain lies along these dimensions will affect the requirements of a system for critiquing in that domain.

2.3.1 Miller's Two Dimensions

The first dimension is the degree of *independence of decisions.* Treatment plans in a domain such as anesthetic management can be critiqued without a global analysis, since actions are independent: the choice of intubation method is unrelated to the drug chosen to induce anesthesia. On the other hand, in a domain like multiple trauma, the best way to handle a stab wound depends on the complete set of goals currently being pursued, including those arising from other wounds. Here, a *global evaluation* of the plan is important.

The second dimension is the *depth of domain knowledge* required to produce an effective critique. The most straightforward domains for critiquing are those in which there are established approaches to management, such as the oncology protocols used in ONCOCIN [21]. Here, the critiquing system needs only to recognize where the user's solution deviates from the established procedure. At the other extreme are problems that require reasoning from basic principles. Miller also identifies an intermediate level of domain knowledge that he refers to as the level of *treatment goals.* In the domain of ventilator management, for which there is no standard protocol, his VQ-ATTENDING system identifies a set of treatment goals

from the patient description, using straightforward production rules. It then bases its critique on how well the physician's treatment plan addresses these goals.[1]

The critiquing model I am developing is designed to handle domains in which decisions are interrelated, requiring global plan evaluation. With respect to the second dimension, my model is goal-directed rather than appealing directly to either standard protocols or basic principles. The trauma domain does not have a standard protocol to handle every situation that might arise.[2] It is, however, possible to formulate an expert's knowledge of trauma management in terms of diagnostic and therapeutic goals and how best to achieve them in different situations, which is the approach that TraumAID 2.0 uses in developing its plans. The level of goals provides an intuitive basis for reasoning about, and critiquing, the management of patients with severe injuries.

2.3.2 Two New Dimensions

My work on critiquing has led me to identify two additional dimensions that seem relevant to the design of critiquing systems. The first is the *amount of time* between plan development and plan execution. Unlike most critiquing systems, TraumAID functions in a domain in which parts of a plan may be executed almost as soon as the plan is developed. This means that:

1. The user does not have a great deal of time to attend to a computer-generated critique.

2. The system must be able to draw inferences based on incomplete knowledge of the situation and the physician's plan.

The second new dimension that can influence critiquing system design is the extent to which *preferences* are involved in decision-making. Where a domain lies on this dimension affects the amount of specific decision-making knowledge that is required for critiquing. Several researchers have pointed out that one of the advantages of critiquing is that it does not require a complete specification of the domain in order to be able to critique proposed solutions [11]. This does not mean, however, that such a specification is not *useful*, but rather that if it is not available, a critiquing system can still be implemented since it is possible to critique a proposal without having an alternative plan worked out. In some domains, such as the domain of kitchen design considered by the critiquing system JANUS [11],

[1] Plan inference in this system is simple because the goals he considers for ventilator management are independent, and each action can only be used to satisfy one goal. Hence there is a direct relationship between the physician's treatment plan and the underlying treatment goals.

[2] This is not the same as having standards for decision making, which do exist in trauma management: e.g. goals are prioritized on the basis of logistic constraints, cost minimization (embodied in both staged diagnosis and local procedure preferences), and according to the "ABC's" of trauma care, depending on whether they address problems of the airway, circulation, etc.

almost all decisions are based on preferences, and the critiquing process basically involves making sure that certain constraints are met. In such a domain it would be unnecessary for the critic to be able to make specific design decisions (such as exactly where in the kitchen to place the stove) independently of the user. Thus, the knowledge representation required for such systems is just a specification of the relevant constraints.

On the other hand, in the domain of multiple trauma (as in other medical decision-making domains) few decisions are purely a matter of personal preference. Rather, there are guidelines and strategies that have been developed by larger or smaller communities as being the way in which medicine will be practiced. Critiquing in these domains is a combination of enforcing hard constraints and making the physician aware when the plan he is proposing is significantly worse than "optimal." A critiquing system with these requirements must have access to a knowledge base that specifies as closely as possible the "best" things to do in a particular situation. It makes sense, therefore, in this type of critiquing system for the critic to develop its own solution to the problem, and to compare it to the solution proposed by the user. Non-critical decisions can be left alone simply by choosing not to comment if the user's plan differs from the system's on these points.

Chapter 3

Related Work on Critiquing

While the term *critiquing* was first used to describe systems that evaluated medical treatment plans, it has since been applied to a wide range of knowledge-based applications. The basic approach of providing an evaluation of a user's solution has proved to be of use for a variety of problems. As I suggested in the previous chapter, the design of a critiquing system depends to a considerable extent on the features of the task domain in which the system functions. In this chapter, I present a brief review of some of the literature relevant to critiquing and discuss their contributions and weaknesses.

3.1 The ATTENDING family of critics

The original ATTENDING system [27], which was the first to be called a critiquing system, was designed to critique the management of anesthesia for patients undergoing surgery. Miller was motivated to introduce the critiquing approach by a desire to develop expert consulting systems that would be useful in domains characterized by subjectivity and variability in treatment practices. Many areas of medicine have the feature that there is not always one "correct" way to do things. A physician's choice of procedure may depend on her own personal experience or on nuances in the patient that are difficult to quantify in a knowledge base. Traditional expert systems that produce a solution to a problem input by the user are not well equipped to handle such situations. First, while their solutions may be "good" in a general sense, they may not always be the "best" solution under the circumstances because they do not take the physician's subjective preferences into account. Furthermore, a consulting system that gives the impression of telling the doctor what is the "right" thing to do may, on the one hand, lead to the doctor taking that advice too literally without first giving it a careful evaluation, or on the other hand it may lead to frustration and disuse of the system if it is constantly presenting the doctor with solutions that he does not agree with.

The original ATTENDING uses a system of rules implemented as Augmented

Decision Networks (ADNs), based on the model of Augmented Transition Networks (ATNs), to evaluate the user's proposals for anesthetic management. The ADNs represent choices that must be made about a patient's anesthesia in a hierarchical structure that captures relationships between decisions and subdecisions. Arcs in the ADN are connected to Problem Management Frames that indicate the anesthetic implications of certain medical problems. ATTENDING uses its knowledge of anesthesia and risks associated with various conditions to evaluate the physician's proposed approach and suggest appropriate alternatives. Finally a prose generator based on ATNs produces a critique of the physician's solution including a discussion of the risks associated with various techniques and confirmation of decisions it finds acceptable.

Further investigation of the critiquing approach led to the development of a family of critiquing systems based on the ATTENDING model. The motivation for developing these prototype systems was, as discussed in Chapter 2, to explore the critiquing approach in different domains in order to identify features of domains that would lend themselves to critiquing, and to explore different facets of critiquing applications. A complete description of the ATTENDING family of critiquing systems is presented in Miller's book [28].

HT-ATTENDING is a critic for the pharmacologic management of essential hypertension. This domain has the properties that there are a huge number of drugs that can be used to treat a patient with hypertension, and that clinical knowledge in the field is in rapid flux. Miller envisions a critiquing system for this domain as a sort of electronic survey paper that is capable of explaining to the physician how his particular style of treatment fits in with current thinking in the field.

VQ-ATTENDING critiques ventilator management for patients on mechanical respiratory support. This domain is interesting because it lends itself to a form of *goal-directed critiquing.* In designing a critiquing system for ventilator management, Miller found that it was useful to separate the system's *strategic* knowledge about management goals from the *tactical* knowledge about how to achieve those goals. The goal-directed approach seen in VQ-ATTENDING is somewhat similar to my approach to critiquing trauma management. However, it is not able to handle interacting goals or to critique the choice of goals as well as the management choices. Furthermore, it does not function on-line, during the ventilation process, but rather critiques the entire ventilator management plan prior to its execution.

PHEO-ATTENDING is a system for critiquing the laboratory and radiologic workup of patients for suspected pheochromocytoma (a rare kind of tumor of the adrenal gland). Workup is the process of performing tests and procedures in order to rule in or out a particular diagnosis. Optimizing workup is important because the tests and procedures involved can be costly. In TraumAID, this kind of diagnostic activity is treated in a goal-directed manner, along with the planning of therapeutic activity.

From the problem of workup, Miller moved to critiquing differential diagnosis in

radiology with the ICON system. With ICON, a radiologist enters a set of findings from a chest x-ray and proposes a diagnosis in response to these findings. The system produces a discussion of how the findings relate to the proposed diagnosis, and what additional findings might help to clarify the diagnosis further.

Finally, Miller has developed a shell, E-ATTENDING, for the development of critiquing systems that includes a differential analyzer and a prose generator.

Miller has thus developed systems that start with a diagnosis and critiques proposed therapy, and a system that starts with findings and critiques proposed diagnoses. He does not, however, take the step taken in TraumAID of combining diagnosis and therapy and critiquing an integrated management plan. In part, this is a function of the domains he has chosen to work with, which do not involve pressure to make decisions and act quickly to address urgent problems. In a domain like trauma management it is necessary to interleave diagnosis and therapy so that problems can be addressed within a limited time frame. In less time-critical domains it is possible to carry out a complete differential diagnosis before beginning to plan appropriate therapy. Also, the focus of these systems on getting a correct diagnosis before proceeding to treatment is not shared by TraumAID. It is less important to TraumAID that the physician gets the correct diagnosis than that he *does the right thing.*

3.2 ONCOCIN: User-guided critiquing

The ONCOCIN system [21] is an example of an expert consulting system that was adapted to critique user solutions rather than present its own recommendations. In the course of testing their original system, the developers of ONCOCIN found that its users had significant difficulty with the fact that they had to justify their reasoning each time they wished to override the system's decisions. In response to this complaint, the developers decided to move to a critiquing interface that would allow the user greater autonomy in the decision-making process, and would therefore have a positive effect on system acceptability. The knowledge base in ONCOCIN was based on existing protocols for cancer treatment, but there are cases where the physician may wish to vary from the protocol, and the critiquing system allows for such deviations without requiring an irritating override procedure.

To convert ONCOCIN from a traditional consultation system to a critiquing system, a hierarchical plan analyzer was added. The program still develops its own solution to the patient's management, but rather than present this solution to the user immediately, it waits and prompts the user to enter her proposed management plan. The user's solution is then formatted into a hierarchical structure and compared step by step to the "correct" solution derived by the program.

The critiques produced by ONCOCIN are in the form of a user-guided explanation of the system's reasoning. After the plan analyzer produces a list of the significant differences it finds between the user's plan and its own, it presents these

differences to the user and asks if he would like further explanation of how ONCOCIN reached its conclusions. To explain how a conclusion was reached, ONCOCIN presents an English translation of the rule that caused it to form that conclusion. The facts that support that rule are then added to the *agenda* – a list of items that the user may want to know more about. At each point during the critiquing process the user is presented with the current agenda and asked which item he wishes to have explained. In this way, rather than presenting *all* the relevant information or using a user model to determine what information the user might like to see, the critique is tailored to his specific interests.

While this method of presentation allows the user to look at only those items in which he is specifically interested, it has the disadvantage that he must actively elicit the critique using a process that is somewhat awkward and time-consuming. This may be acceptable in the context of a pre-treatment consulting session in which ONCOCIN was designed to be used, but it would not be feasible for a system that is intended to function during the management of a patient in need of urgent medical attention.

3.3 Critiquing from Automated Medical Records

The HyperCritic system developed by Van der Lei [48] looks at patient data stored in automated medical records of patients with hypertension and critiques the therapy reported in those medical records. HyperCritic identifies significant events in the medical records and then uses a set of domain-independent *critiquing tasks* to assign critiquing statements to those events. From the point of view of system design, the main contribution of the HyperCritic system is the separation of domain knowledge from critiquing knowledge. Unlike the decision rules in the ATTENDING family of systems, HyperCritic's four categories of critiquing tasks – preparation, selection, monitoring, and responding – are not stated in terms of specific medical knowledge, but rather in terms of more general properties, such as side effects, contraindications, and appropriate dosages. For example, one of the selection tasks is triggered whenever a new drug is started, and checks to see if any contraindications for that drug are present in the medical record. Specific drugs and contraindications are not mentioned in the specification of the critiquing task. In his thesis, Van der Lei shows how this separation of domain knowledge and critiquing knowledge facilitates acquisition and maintenance of the medical knowledge base in the HyperCritic system.

Another important contribution of Van der Lei's work is a thorough analysis of the feasibility and effectiveness of critiquing from automated medical records in the domain of hypertension. He compared the critiques produced by his system to critiques produced from the same medical records by eight physicians. He found that there were several areas where the system failed to produce comments that were produced by the physicians, and on the whole HyperCritic tended to be less

critical than the physicians in terms of the number of comments it produced. However, when the system did produce a comment it was quite highly correlated with the physicians' opinions. The evaluation indicates that the critiquing system *can* produce useful comments from the data in medical records.

3.4 Critiquing Designs

Another area in which the critiquing paradigm has been applied with some success is that of design critiquing [12, 19, 39]. This is a rather different application of critiquing than I am concerned with for this project. Critics have been implemented to assist with the design of buildings, individual room layouts, computer programs, etc. In these programs, the system is provided with a set of rules or constraints for evaluating a design. When the proposed design violates any of these constraints, a critique is generated. One advantage of this type of critic is that it can be embedded as part of a computerized design environment, so that the design being critiqued is directly available to the critiquing system and there is no concern with modelling external phenomena within the system. In addition, since there is less of a sense of there being a "right answer" in design critiquing than in plan critiquing, the system does not have to generate its own solution in order to produce a critique.

The Archie system [19] for assisting design in the domain of architecture uses a case-based approach to critiquing. The user interested in evaluating a potential building design inputs a description of the conceptual design of the building, and the system searches its database for relevant *stories* taken from evaluations of existing buildings. These stories serve as a critique, pointing out potential problems with the design the user has suggested.

Another approach to design critiquing appears in LISP-CRITIC [11]. This is a system that helps programmers to improve their lisp code by "suggesting transformations that will make the code more cognitively efficient (i.e. easier to read and maintain) or more machine efficient (i.e. faster or smaller)." [1] LISP-CRITIC generates its critiques on demand and allows the user to accept or reject its suggestions. One shortcoming of this system is that it is not able to evaluate the effect of its suggested transformations on the *correctness* of the code.

JANUS [11] is a design environment for the construction of kitchen designs. The critiquing component of JANUS has knowledge about building codes, safety standards, and functional preferences. When a rule is violated, the system displays a message explaining the nature of the problem. An extension of this system, KID [13] allows the user to specify their high-level goals and priorities for a particular design, thus introducing greater flexibility into the system. The KID system is user-extensible, allowing its users to add to the rule-base if the specific situation they

[1] This is similar to the Program Enhancement Advisor described in [31] except that the latter is more concerned with the knowledge representations necessary to explain the system's reasoning.

are concerned with is not covered.

3.5 The deep generation of critique text

As I have mentioned previously, one of the major motivations for developing critiquing systems over more traditional consulting systems is the potential for increased acceptability to the user. Therefore, it is important that the output of these systems be easy to understand. To this end, Rankin [38] has investigated methods for the deep generation of text in critiquing systems. Deep generation, as opposed to surface generation, is concerned with establishing the *content* and *ordering* of the comments that will make up the critique. Surface generation – finding the actual words and phrases to express the content specified during deep generation – is not addressed in this work.

Rankin's investigation of critique generation was motivated by Miller's work with ATNs and expressive frames to generate text. He first examines these techniques in the context of an experimental Miller-type implementation called CRIME (CRitiquing In MEdicine), a system for critiquing the treatment of urinary tract infections (UTI). This is a simple domain, with a small number of decision points and a fairly standard protocol for treatment, so the number of comments that the critic needs to produce is small and the choice of appropriate critiques is straightforward. While it was possible to develop an ATN to produce reasonable critiques in this domain, Rankin concludes that this approach is inflexible and not suitable for large-scale applications or unpredictable domains. Another criticism he makes is that the production of critiques in these systems does not take the individual user into account, producing the same comment regardless of who it is addressing and what that person might already know.

Rankin has developed a more general and reusable approach to deep critique generation involving three stages. First, establish the expressive goals of the critique. Second, determine the exact content of the critique based on the specific situation and knowledge about the user. Finally, put the critique comments in an appropriate order. He first considers how to determine the goals of the critique. In the domain of UTI treatment, he identifies four main types of comment that should be available: the system can CONFIRM correct decisions, INFORM the user of information he may not be aware of, WARN the user when a proposed treatment presents some risk to the patient, and JUSTIFY[2] its conclusions by explaining its reasoning. The first step in critique generation in Rankin's model is to examine each part of the user's proposed treatment, generating comments corresponding to these four types wherever they are appropriate. For example, when the user proposes a treatment that the system finds to be correct, a CONFIRM comment is gener-

[2]Justification is actually a set of sub-types including elaboration, motivation, cause, and sequence.

ated. When some additional information is found to be relevant to a decision, an INFORM is generated along with a JUSTIFY to explain the underlying reasoning. All these comments together represent the goals of the critique.

The next step in the process is to determine the final content of the critique by eliminating redundant or unnecessary comments. While all the comments produced in the first stage represent communicative goals, some of these goals may have already been achieved, some of them may be implicitly understood, and some of them can be assumed to be part of the user's beliefs already. To tailor the critique contents to specific users, Rankin's system has a user model that keeps track of a representation of the user's beliefs at every point during the consultation. For example, if the user prescribes an antibiotic, Trimetoprin, to treat a patient's infection, the user model will be updated to include the belief that the patient should be treated with an antibiotic, *and* the belief that the antibiotic should be Trimetoprin. When the system informs the user of a fact, the user model is updated to include a belief about that fact, and if the new belief conflicts with a belief already in the user model, the old belief is removed to maintain the consistency of the model. The user model is used to determine which comments are relevant and should be shown to the user. Information that the user already believes (according to the user model) is not included in the critique.

The final stage in deep critique generation is to organize the comments into a reasonable order. To produce a coherent critique, it is desirable to link related statements together in the final output, either by juxtaposition or by marking certain relationships such as elaborations or motivations with cue words. Rankin uses Rhetorical Structure Theory (RST) [25] to organize the comments in his critiques into longer sequences. RST is a model of text structure that uses predefined *schemas* to represent relationships such as elaboration, motivation, and causation, that can exist between different portions of text. Rankin uses RST schemas to link statements to their justifications in appropriate ways. The schemas specify the order in which statements should be given, as well as appropriate choices for cue words and phrases such as "because" in an *elaboration* schema or "may result in" in a *cause* schema.

While his model does not have anything to say about plan analysis in complex and unpredictable domains, Rankin makes some interesting points about how to generate relevant, coherent, and useful critiques. His first insight is that a general procedure for producing text can be initiated by classifying statements according to the goals they are supposed to achieve. Another important point brought out in this paper is that communication with a user can be facilitated by building and maintaining a model of that user's knowledge and beliefs. Finally, Rankin demonstrates how relationships between statements in the critique can be reflected in the ordering of the text.

THEN these error types are likely	IF these symptoms exist
Information Acquisition	
Availability	Using only easily available information and ignoring not easily available sources of significant information
Base rate	Ignoring abstract information at the expense of concrete information
Information Processing	
Adjustment and anchoring	Using heuristics which may reduce the mental effort required to arrive at a solution at the cost of using the full amount of information
Representativeness	As sample size is increased interpreting the results of small samples to be representative of the larger population
Intended Output	
Fact-value confusion	Regarding and presenting strongly held values as facts
Wishful thinking	Choosing an alternative that one would like to have associated with a desirable outcome
Feedback	
Ease of recall	Being affected by data which can easily be recalled or assessed
Fundamental attribution error	Associating success with personal inherent ability and failure with poor luck
Knowledge	
Missing knowledge	Trying to solve problems, make decisions or form plans in multi-discipline domains

Figure 3.1: Some rules for identifying common biases

3.6 Critiquing Based on Cognitive Biases

In his research on developing automated critics, Silverman [43] has developed what he calls a generative theory of "bugs" to produce a rule base of errors resulting from cognitive biases in information acquisition, information processing, intended output, and attention to feedback (see Figure 3.1). He also recognizes incorrect or missing knowledge as another potential source of error. This rule base is used as a basis for evaluating the domain of the critic and determining which types of errors are likely to occur. Once the typical sources of error have been identified, appropriate critiquing strategies can be developed to inform the user about his errors.

In the application presented in [43], forecasting potential threats for Army equipment during missions, the user's input is restricted to simple answers to queries from the system. The system's explicit knowledge of judgment biases, along with some

superficial knowledge about when and where these biases may appear in the current domain, allow it to warn the user when he is exhibiting judgment characteristic of any of these biases. The system does not have any deep knowledge of the domain, or of the user's knowledge or reasoning processes. No reasoning about plans is involved in this critic.

Ongoing work by Silverman and his colleagues covers three related lines of investigation [45]:

1. Expanding the rule-base for the identification of biases to include a greater number of lower-level rules.

2. Identifying the occurrence of biases in the behavior of professionals in specific domains.

3. More in-depth studies of how to identify and critique a single bias, such as confirmation bias.

3.7 The Role of Explanation in Critiquing

In their analysis of the attitudes of physicians toward computer-based consultation systems, Teach and Shortliffe [46] found that while doctors do not demand that a consultation system always be *correct*, it must be able to *explain* its decisions. In developing explainable expert systems, researchers have been concerned primarily with such issues as the level of knowledge representation necessary to provide good explanations, and how to organize explanation into a coherent text [31]. More recently, more sophisticated approaches have been proposed, integrating work from the natural language generation community with work on explainable expert systems [32], and allowing for interactive explanation, where the user can request additional clarification or justification in response to the system's explanations [30].

In [29], Mittal and Paris discuss the differences between text generation for explanation and for critiquing, both in terms of surface (tactical) generation and deep (strategic) generation. In critiquing, the interpersonal aspect of the interaction is much more important than it is in explanation. With respect to surface generation, they point out that because of the different roles of explanation and critique, phrasing in an explanation is not as important as it is in a critique. Since a critique is an evaluation of the user's reasoning, it is important to present it in a manner which will not be insulting or argumentative. An understanding of the impact of particular speech acts, words, and phrases is important for this purpose [3].

Deep generation involves selecting the content and organization of text. Both of these elements differ between explanation and critiquing. In terms of the content of the text, unlike explanations, critiques need to present alternative solutions to a single problem. In terms of the structure of the text, Mittal and Paris point out that the rhetorical relations that appear most often in critiques, such as *concession,*

exhortation, contrast, antithesis, and *justification*, differ from those that tend to appear in explanatory texts, such as *background, attributive,* and *elaboration.*

In designing a critiquing system, it is important to keep in mind the specific role of the text it produces, using a critiquing style that is appropriate to the particular audience that will be using the system. This issue will be explored further in the discussion of critique generation in Chapter 4.

3.8 Reminders and Alerts

Another approach to on-line decision support that is related to critiquing is represented by reminder or alerting systems [4, 26]. These systems are designed to monitor the data stored in computerized hospital information systems and produce alerts or reminders when a situation arises that should potentially be attended to. The knowledge in these systems consists of long lists of rules that when satisfied will produce an alert or reminder.

While studies have shown that these systems can be incorporated into the hospital environment in such a way that health care providers will acknowledge them, they are often not acknowledged until a significant amount of time has passed. Furthermore, even when an alert is acknowledged, it is not clear how often it is acted upon.

Another drawback of these systems is that they do not have an inference engine capable of accomodating interactions between different medical conditions. This means that they can produce conflicting or irrelevant advice if an interaction is present that is not accounted for *a priori* in the individual rules. Furthermore, they may fail to suggest an action that would be appropriate given a combination of two conditions, but not if only one of those conditions were present.

Finally, these systems differ from the critiquing model in that they do not respond to the intended actions of the physician. An alert or reminder will be provided regardless of whether the physician has already indicated an intention to address that issue. Therefore, these systems are bound to produce a number of unneccessary comments, which may be the reason that physicians using the HELP system [4] indicated that the best method of communicating alerts would be to relay them to a nurse, who could then evaluate them and pass them on to the physican, thus adding a level of indirection between the system and the primary care-giver.

Chapter 4

TraumaTIQ: a Real-Time Critiquing Interface for Trauma Care

In designing an information-delivery system to provide decision support during trauma care, it is helpful to look at some of the characteristics of the trauma situation and how they reflect on the information needs of the physician. In fact, the approach that I have taken in designing this system could be applicable in any domain that exhibits these features:

1. **Multiple goals**: Trauma management often involves reasoning about multiple interacting goals. Therefore, the physician's proposed actions must be evaluated globally, in the context of the entire plan. Additionally, the presence of multiple goals with varying degrees of urgency means that the critique must address not only what actions are performed but also the order in which they are carried out.

2. **Time constraints**: In trauma management, diagnosis and treatment of different actions are carried out within the same time frame, and the urgency associated with actions makes it important to intervene within a short period of time. The critiquing system must therefore be capable of constantly updating its representation of the situation and the user's plan, and revising its critique based on new information.

3. **Task-centered activity**: In an emergency situation such as trauma resuscitation, the physician's primary focus of attention is reserved for the task of caring for the patient. Therefore, the amount of work that he has to do to understand the critique should be kept to a minimum. The system should avoid saying anything that (a) the physician already knows, (b) is "common knowledge" to anyone who would be using the system, or (c) reflects trivial differences between the system's preferences and the physician's.

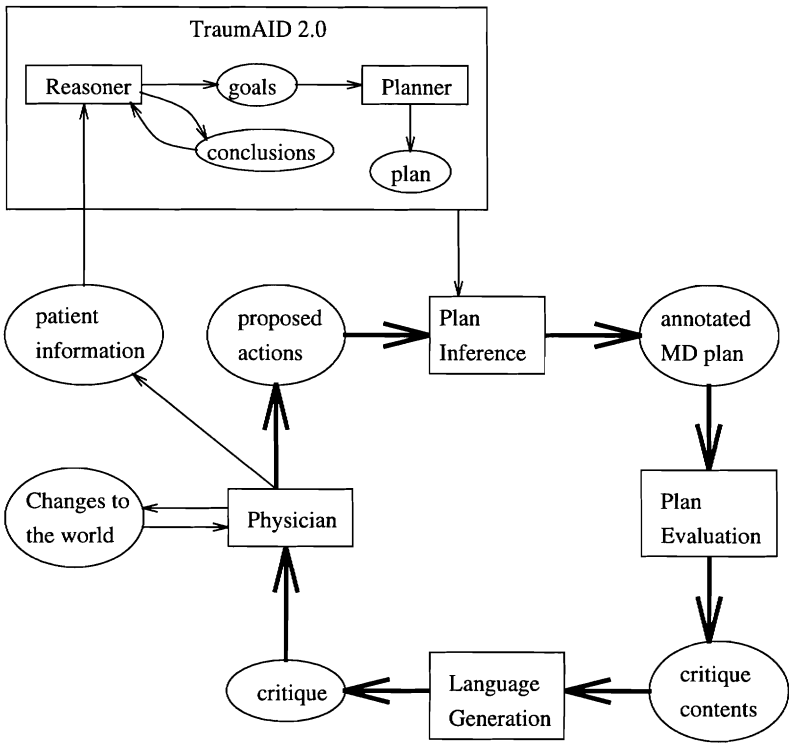


Figure 4.1: The TraumaTIQ module

4. **Gap between order and execution**: Actions that involve resources that need to be brought to the trauma bay or that can only be done elsewhere must be *ordered* prior to being carried out. Since orders can be rescinded, comments pointing out problems with an order can potentially make a clinically significant difference to patient management.

In this chapter I will describe TraumaTIQ, the system I have implemented for critiquing trauma management plans. In contrast to the critiquing systems presented in Chapter 3, which produce their critiques off-line during a consultation session with the physician, this is a process-oriented approach to critiquing: rather than presenting *one* critique based on a complete specification of the problem and the proposed solution, this approach produces critiques in parallel with planning and execution. The critique is continually updated as the information available to the system changes. TraumaTIQ takes advantage of the fact that many actions require resources to be brought to the emergency room or must be done elsewhere, and these actions must be *ordered* ahead of time. Since orders can be rescinded, a well-timed critique could prevent an inappropriate order from being carried out.

The critiquing process is triggered whenever a new piece of relevant information is made available to the system. This information can be in the form of (1) bedside findings, (2) diagnostic test results, (3) therapeutic actions performed, or (4) actions ordered by the physician. Critiques are generated based on the complete set of

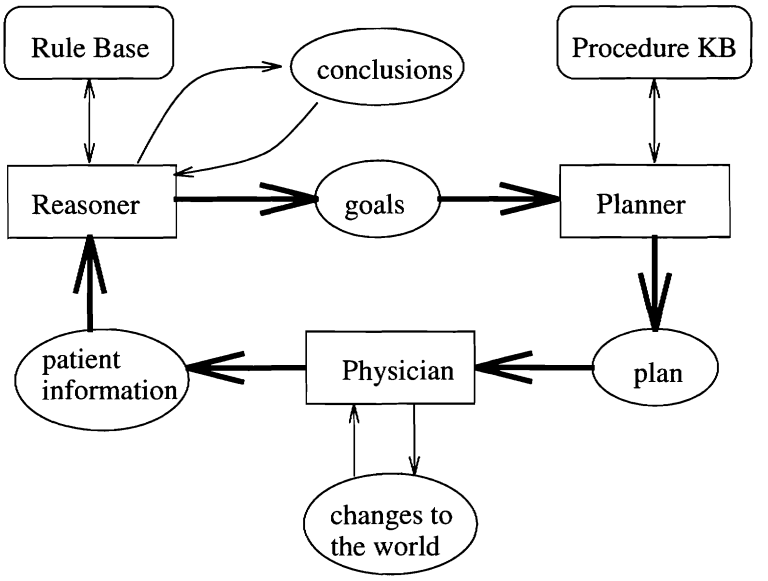


Figure 4.2: System Architecture of TraumAID 2.0

orders that are pending at a given time, so that pending orders that were previously accepted as appropriate may later be critiqued on the basis of new evidence. Once an action has been done, however, it will no longer be considered as on object of the critique, whether or not it was appropriate.

The TraumaTIQ module comprises three central components: the plan recognizer, the plan evaluator, and the critique generator (see Figure 4.1). The design of these components is described in detail in the following sections. I will introduce them in turn and discuss how they address the problems mentioned above. The chapter begins with a brief discussion of the architecture and knowledge representation in TraumAID. For a more complete discussion of the TraumAID system, see [40, 41, 49].

4.1 An Overview of TraumAID 2.0

At the core of the TraumAID system is the integration of a rule-based reasoner that reasons from evidence to conclusions and management goals, with a planner that determines how best to satisfy the set of currently active goals (see Figure 4.2). Using this approach, TraumAID is able to flexibly interleave diagnostic and therapeutic action, thus addressing the competing needs arising from multiple injuries.

The role of the reasoner is to determine a complete set of currently relevant goals for the planner to address. To do this, it makes use of two types of rules: *evidential rules* that map from evidence (observations, test results, intermediate conclusions, and information about the performance of actions) to conclusions, and *goal-setting rules* that map from evidence and conclusions to diagnostic and therapeutic goals.

Whenever any new evidence is introduced, the reasoner is triggered, forward chaining through its entire set of rules and posting the results of any goal-setting rule that fires to the list of currently pending goals. After the new set of goals is complete, the planner is invoked to determine how best to satisfy this particular combination of goals.

TraumAID's reasoner controls information acquisition using a conservative, *staged* strategy for diagnosis and treatment [41]: expensive, definitive tests are not included in a plan until they are justified by less costly tests or observations, and definitive treatment is not recommended without the results of sufficient evidence from diagnostic tests.[1] These strategies are reflected in the knowledge base by the occurrence of related management goals, such as a goal to diagnose hematuria (blood in the urine), which if present, triggers a goal to diagnose bladder injury, which in turn can lead to a goal to treat bladder injury. On the other hand, goals that do not participate together in a coherent strategy may still be connected by test results. For example, the goal of finding a bullet in the mediastinum leads to doing a lateral chest x-ray. While this might also reveal a fractured sternum, there is no strategic relationship between the goal of finding a bullet in the mediastinum and the goal of treating a fractured sternum.

The planner's knowledge base represents the relationships between three different concepts: *goals*, *procedures*, and *actions*. A single goal may have a number of alternative procedures that could be used to satisfy it. Furthermore, a procedure may be applicable to more than one goal. Procedures comprise sequences of actions to carry out and/or sub-goals to achieve. The planner has rules for mapping goals onto alternative procedures, and procedures onto action sequences.

The choice of procedure to satisfy a goal is affected by a combination of local preferences and global optimizations. The order of procedures within a goal-procedure mapping reflects the local preference for individual procedures. That is, if no other goals were currently influencing the decision, the first procedure given would be the preferred one. However, the choice of procedure is also influenced by other active goals. If a less preferred procedure can be used to satisfy two goals at once and result in a more efficient plan, that procedure will be chosen in place of the locally preferred one.

The planner initially uses a "greedy" algorithm that iterates throught the current set of goals ordered by importance, selecting a procedure to address each one not already addresesed by an already selected procedure and then ordering those procedures with respect to each other according to relative urgencies, logistical constraints, standardized priorities, and approximate temporal extent. After this initial planning phase, the beginning of the plan is optimized to ensure that the plan takes advantage of all procedures that can be used to satisfy more than one active goal.

[1]The one case where this is not true is when a patient comes in near death, with catastrophic chest wounds. Surgery is recommended immediately, without attempting to diagnose what specific injuries may have been sustained.

Only the beginning of the plan is optimized because complete optimization is NP-hard and, furthermore, may be wasteful since later parts of the plan may change when new information is introduced.

4.2 Input and Output

Before going on to discuss the architecture of the critiquing module for TraumAID, I will briefly describe the physical set-up of the system in the Emergency Center and explain how information is input and output.

The TraumAID system is designed to run on a Macintosh computer operated by a trauma nurse in the emergency center. The interface is designed as an electronic version of the Trauma Flow Sheet (TFS), a paper form which is the standard for recording information during a trauma case. All data that are relevant to the case are entered into this interface by a *scribe nurse*. These data include observations, test results, monitoring data, drugs administered, etc. In addition, when a procedure is *ordered*, indicating the need for equipment to be set up or brought to the bedside to perform that procedure, the order is recorded in the system. A certain amount of lag time generally occurs between the time a procedure is ordered and the time it is carried out, in order to set up the necessary equipment. In the meantime, the critiquing module can process the information about ordered procedures to infer and evaluate the physician's proposed plan and produce the relevant commentary.

In addition to the main console of the computer, an output monitor has been designed to be placed in the view of the physician heading the trauma team. This monitor will contain any relevant information that the physician might be concerned with, including vital statistics, information about the patient (such as medication allergies), and proposals and/or critiques from the system, along with requests for information that TraumAID believes to be important. In the future, synthesized speech with contextually appropriate intonations may be used to convey the critique to the physician, thus obviating the need for him to look up to read output on the monitor [36]. Alternatively, the critique may be presented to the scribe nurse, who can then convey it to the physician.

4.3 Recognizing the Physician's Plan

From the early days of plan recognition research, it has been recognized that understanding a user's goals or intentions is important if systems are to be able to respond intelligently to the user's needs [1, 42]. This idea has been exploited in such applications as dialogue and text understanding [1, 5, 6, 15, 22], intelligent computer-assisted instruction (ICAI) [23, 24], and intelligent interface design [14]. At the same time, formal models have been developed in order to study the semantics of plan recognition in a rigorous way [18].

Plan recognition problems vary according to a number of properties (see [18]). First, can the inferring agent assume that the planning agent wants his intentions to be inferred (*intended recognition*) or not (*keyhole recognition*)? Second, does the inferring agent have complete knowledge of the domain about which it is reasoning? Third, is it possible that the plan being inferred may be erroneous in some way? The following discussion represents two approaches to plan recognition that have appeared in the literature, reflecting different combinations of the above features.

4.3.1 Approaches to Plan Recognition

Inferring Plans in Cooperative Dialogue

A great deal of the literature on plan recognition has been concerned with the problem of providing cooperative responses in dialogue. The early work in this area drew from the more established work on classical planning, using a representational framework modeled after the approach presented in STRIPS [10]. In the STRIPS formalism, a plan is a sequence of *operators* leading from a starting state to a goal state. A plan operator consists of an action description called a *header*, a list of *parameters*, a set of predicates that represent *preconditions*, and an *add list* and a *delete list* that represent the action's effects. Classical planning systems start with an initial state of the world and a goal state and determine a sequence of operators that together would have the effect of bringing about the goal state.

Plan-recognition systems based on this representation compare the observed actions of an agent to the operators in a plan library and determine the potential plans of which those actions could be a part. These systems work on the assumption that if an agent is understood to have a plan that could be part of some higher level domain plan, then the agent is actually pursuing that higher level plan. In language understanding systems [1, 5, 42], the observed actions are utterances, which are assumed to fit into an overall plan on the part of the speaker. The recognition of domain plans is recursively generated from the recognition of utterance-level intentions or speech acts using heuristic rules to determine the most coherent relationship possible. The plan structure can then be used as a context for the interpretation of future utterances, as well as a means to determine appropriate responses.

More recent work in this area has focused on the need for more flexible representations of plans to account for such phenomena as reasoning about plans that have not yet been adopted, constructing alternative plans to achieve the same goal, and recognizing incorrect plans [20, 22, 34]. In particular, Pollack [34, 35] has argued that a plan should be thought of not in terms of a static recipe-for-action, but rather in terms of the complex mental attitudes that people have toward the actions that make up their plans. In her view, a plan is a collection of beliefs about the world and intentions to perform certain actions. Thus a plan recognizer must infer the speaker's beliefs and intentions in order to evaluate them and correct any misconceptions that they might reflect.

This literature is interesting from the point of view of critiquing, since it has a lot to say about detecting and correcting misconceptions in a speaker's plan. However, since they are concerned with dialogue understanding, these models operate under an assumption of intended recognition. They generally assume that either the user's goal is given as part of his query or it is uniquely identifiable from the actions in the query. In a critiquing application, on the other hand, while it is important to be able to recognize incorrect plans and to correct misconceptions, it cannot be assumed that the user will make his goals clear to the system.

A Formal Model of Keyhole Recognition

In his work on plan recognition, Kautz [18] focuses exclusively on keyhole recognition of correct plans where the system has complete knowledge of the domain. He represents knowledge of plans as an abstraction hierarchy, connected by is-a links and is-step links. The plan inference process involves generating and refining a plan graph representing the agent's plan until it can be described in terms of a top-level plan, or end-event. Since he is doing keyhole recognition, the user's top-level goals are not known beforehand.

When the plan is ambiguous, i.e. there is more than one combination of top-level goals that will explain the observed behavior, Kautz takes the approach of looking for the plan with the least number of "end events" or unrelated goals. This solution is based on the assumption that the simplest plan is most likely to be the actual plan. This assumption does not apply in trauma management, however, since it is often the case that (1) there are multiple, unrelated goals, all of which may require action, and (2) one action can be used to address a number of different goals (for example, a single X-ray can be used to diagnose a number of injuries in one area of the body). It would be inaccurate in such a domain always to assume that the *simplest* plan (i.e. the plan with the smallest number of high-level goals needed to explain all the observed actions) is the one being followed. A more realistic approach in this case is to use information about the situation in which the plan is being developed in order to infer the *most likely* plan that the physician would have developed that explains all the observations.

4.3.2 TraumaTIQ: Using expectations to infer a plan

As Goodman and Litman [14] point out, formal models of plan recognition, while interesting in their own right, tell us little about how to implement a *practical* plan recognition system. Practical system design often involves making assumptions about the user and the domain in order to control computation and resolve ambiguities. These assumptions can often take advantage of special features of the domain and the intended use of the system.

The plan recognition problem for TraumaTIQ is an example of keyhole recognition, meaning that the user (in this case, the physician) cannot be treated as a

cooperative provider of information about her plan for managing the patient. What can be observed through the "keyhole" are *actions* that have been performed, and *orders* the physician has placed for actions she wants carried out. One problem this poses is that orders will not necessarily be given and recorded in the system in the order in which they are intended to be performed. TraumaTIQ therefore cannot make the assumption that consecutive orders are likely to be related (i.e. addressing the same or similar goals), as is done with utterances in dialogue understanding systems.

Another factor influencing the plan recognition problem is the fact that multiple goals are involved (diagnosing and treating multiple injuries concurrently), and that these goals cannot be treated as being independent since a single action may often serve to address several goals. This can lead to far more complex plans than have been dealt with in previous plan recognition systems, which generally assume that there is either a single top-level goal, or if there are multiple top-level goals then they are independent of each other.

Furthermore, TraumaTIQ cannot assume that the physician's plans are always correct. Since the number of incorrect plans is too large to encode *a priori*, the system must have some guidelines to interpret orders that do not correspond with its knowledge of possible plans. Neither are the physician's plans complete – since the system is designed to function *during* patient management, plan recognition must be done incrementally, taking into account that at any given time the physician's plan is only partially specified.

In spite of these complexities, however, we have the advantage that the TraumAID system itself should have access to all the relevant information about the situation through the graphical interface to TraumAID, into which the scribe nurse enters data as it becomes available (although the system cannot assume that it has been given all the information known by the physician). Furthermore, TraumAID's reasoning and planning components develop and maintain a complete model of the situation, the goals it considers to be relevant, and the actions it would recommend to be performed at all times. Huff and Lesser have pointed out the advantages of using contextual knowledge and basic domain principles to guide the search for an explanatory plan [16]. The approach I have taken to plan recognition in this project takes advantage of this type of knowledge, together with certain features of the situation in which the system will be used, by making the following assumptions:

- The head of a trauma team will have expert or near-expert knowledge of trauma, and will usually develop plans that are similar to TraumAID's. Thus, if an action has been ordered that is also in TraumAID's plan, TraumaTIQ assumes that it is being done for the same reason(s).

- The physician is more likely to have appropriate goals but be addressing them in a sub-optimal way, than to be pursuing the wrong goals altogether.

- While TraumAID follows a conservative strategy for pursuing diagnosis and treatment from observations, physicians may proceed more rapidly, pursuing a goal that may be involved in a current strategy but for which TraumAID does not yet have enough evidence to conclude its relevance. An understanding of the strategic relationships between goals should help to recognize examples of this difference.

If the system has a goal that it considers relevant, it will try to use that goal as an explanation for the physician's proposed actions. In this way, ambiguities in the physician's orders are reduced, but not eliminated by the plan recognizer. The remaining actions, those that cannot be explained by any of TraumAID's active goals, are left to be clarified with the acquisition of additional information.

TraumaTIQ's plan recognition algorithm works as follows:

1. When an action, α, is ordered by the physician, check whether α is currently a part of TraumAID's recommended plan as a means of satisfying all or part of goal γ, or all or part of each member of a set of goals Γ.

2. If so, add γ or Γ to the representation of the physician's plan.

3. If α is *not* currently in TraumAID's plan, determine whether there is a relevant goal that α might address:

   (a) If any of the goals that might lead to doing α are present in TraumAID's current set of active goals, assume that α is being done to address that goal or goals.

   (b) In the case that there is no relevant goal to explain why the physician is ordering α, check whether any of the possible goals motivating α are part of a currently active diagnostic strategy (cf. Section 4.1).

   (c) If no relevant goal or strategy is found, leave the goal unspecified and add the intention to do α to the representation of the physician's plan with no goal attached. There is one exception to this rule:

   (d) If the system only knows of one possible goal that would lead to performing α, assume that α is being done to address that goal, even though it is not currently relevant.

4.4 Goal-Directed Plan Evaluation

In general, the aim of plan evaluation is to detect misconceptions, lapses in judgment or memory, missing knowledge, and disagreements that are revealed in a plan, in order that they might be corrected. As was discussed earlier in this paper, plan evaluation can be done using a *differential* or an *analytical* approach [11, 12].

The former method compares the user's plan to a "correct" plan generated by the system, while the latter evaluates plans with respect to a predefined specification of constraints on the solution without actually generating its own solution.

One advantage of the differential approach is that it provides a standard on which the system can base its critique. By comparing the physician's plan to a plan that can be assumed to be a broadly acceptable way of approaching the problem, the system has an alternative solution to suggest when it does not agree with the physician's plan. Furthermore, the reasons for choosing a particular course of action can be encoded in the system and used to produce an explanation of the system's behavior. Another advantage of differential evaluation is that it allows a global analysis of the plan. In developing its plan, the planner can detect possible interactions between goals, and find the most efficient way to address that particular combination of goals. A differential evaluation of a user's plan can determine when the user is not reacting to potential interactions between goals.

On the other hand, the analytical approach has the advantage that it does not critique the user with respect to one definitive solution. Rather, an analytical system defines a space of possible plans within which a solution is more or less acceptable. This affords the system some flexibility in dealing with domains where variability and subjectivity are inherent in the decision making process. In addition, the analytical approach has the advantage that the system does not have to be able to generate its own solution to the problem in order to critique a proposal. This makes critiquing possible in domains which are too complex or unconstrained to represent using decision rules.

Finally, while a differential evaluation allows the system to explain why *its* solution is the *right* way to handle the problem, analytical constraints can be used to generate explanations as to what is *wrong* with the *user's* solution. For example, the ONCOCIN critiquing interface used purely differential critiquing, comparing the user's treatment plan to the plan generated according to the system's coded oncology protocols. As a result, the system was able to point out where the user's plan differed from the system's, and could produce, upon request, a translation of the rules that led the system to its conclusions, but it had no capability for explaining what was wrong with the way the user wanted to do things. On the other hand, an analytical system for kitchen design might include a rule that the stove should be no less than five feet from the sink (see the JANUS system [11]). If this rule is violated in a proposed design, the system can cite it as a reason for not accepting the design.

The model of plan evaluation that I am developing for TraumaTIQ makes use of the best features of both differential and analytical approaches. It combines the ability to offer specific advice and to evaluate the plan globally that the differential approach provides, with the flexibility and additional explanatory capabilities of analytical evaluation. In addition, it uses knowledge about the magnitude of different types of errors to filter the output so that only non-trivial errors will be

critiqued. The next two sections describe the model and discuss work to be done in the future.

4.4.1 Identifying errors in the plan

The first stage of plan evaluation in TraumaTIQ is to apply a set of goal-directed evaluation routines. These routines are concerned with errors of:

Omission A goal that TraumAID considers relevant is not being addressed by the physician in a timely manner. This can be further analyzed as to whether:

1. the goal is not being addressed at all, or
2. the goal is only being partially addressed – some but not all the actions in the procedure addressing the goal have been ordered.

Commission An action is present in the physician's plan that *does not* address a relevant goal. If a unique goal can be inferred to explain this discrepancy, that goal can be further categorized as to whether:

1. it has been proven irrelevant by the failure of all TraumAID's goal-setting rules for that goal,
2. it is not proven irrelevant, but there is no evidence that it is relevant,
3. some, but not all, of the necessary evidence to prove that the goal is relevant is known, or
4. it has already been addressed.

Procedure choice A relevant goal is being addressed, but not using the procedure preferred by TraumAID.

Scheduling Actions are not being done in the order recommended by Traumaid, e.g., satisfying urgent goals before non-urgent ones.

The *omission* routine is sensitive to the fact that, due to the real-time nature of the critiquing task, the specification the system has of the physician's plan is likely to be incomplete at any given time. Rather than commenting immediately when a goal TraumAID considers relevant is not addressed in the physician's proposed plan, it notes when a sufficiently long period of time has passed since TraumAID decided a particular goal needed to be addressed without the physician attempting to address it in some way. TraumAID's preferred approach for addressing this goal will also be noted. The amount of time the system will allow to pass depends on the urgency of the goal being addressed. The rule of thumb TraumaTIQ uses is that a comment should be produced after approximately 10% of the time period has passed that is available to address the goal without negative consequences. Furthermore,

an action is not noted as an omission if the underlying goal is *dependent* on any actions that are scheduled to be done before it. In other words, if it is possible that the action will be removed from TraumAID's plan before it is done as a result of additional information, it will not be mentioned as an omission.

The *commission* routine looks for actions that the physician has ordered but that do not have a relevant goal associated with them. If a unique goal has been selected as the reason for ordering the action[2], this routine will seek to determine *why* that goal is not currently relevant. The goal may be irrelevant because it has already been addressed or will be addressed by another procedure that has already been ordered, it may have been eliminated due to the failure of all its rules, or the system may need additional information before it can determine the relevance of the goal. If no unique goal is available, the action is simply noted as unexplained, along with a list of the goals that are associated with that action in TraumAID's knowledge base.

The *procedure choice* routine looks for cases where TraumAID and the physician have chosen different procedures to address the same goal. There are three reasons that TraumAID may have for choosing one alternative procedure over another:

1. The chosen procedure is preferred for addressing the goal.

2. The procedure that TraumAID proposes was chosen as a global optimization in which it was determined that one procedure could be used to address multiple goals.

3. TraumAID may have selected a less preferred procedure due to the presence of a contraindication for the preferred procedure[3].

In any case, when a difference is detected by this routine, the reason TraumAID prefers one procedure over the other will be recorded in the evaluation.

The *scheduling* routine is concerned with enforcing temporal ordering in the plan. Since it is not possible to assume that actions will be performed in the order in which they are ordered by the physician, it is necessary to make some assumptions as to when an error of this type is actually likely to occur. To minimize intrusiveness, the system will withhold its comments if it is *possible* that the correct scheduling is intended. If an action has been ordered by the physician, but other actions that the system believes *must* be done before that action have not yet been ordered, this omission will be noted. However, if both actions have been ordered, the system will assume that the physician is aware of the proper ordering, and will not pursue the matter further.

[2]See the discussion of plan recognition for how this may be done.

[3]Contraindications may take the form of procedure conflicts, which the TraumAID system has in its knowledge base, or patient-specific contraindications, such as allergies. If a procedure choice error appears to be due to the physician not taking a contraindication into account, a simple reminder may suffice for the critique.

4.4.2 Filtering by significance of errors

The first stage of evaluation identifies where TraumAID and the physician disagree as to how best to manage the patient. However, it does not provide any information about the *significance* of these disagreements. For example, the physician may have ordered an unmotivated peritoneal lavage (an invasive test to check for bleeding in the abdominal cavity). This would seem to be a significant error since it has costs in terms of both time and invasiveness. On the other hand, an unmotivated administration of antibiotics would not (unless contraindicated) be considered as important to correct since the costs involved are not high.

The second stage of plan evaluation filters the output from the first stage according to knowledge about which errors are important enough to mention in the critique. The reason for incorporating this knowledge into the system is that it will reduce the number of comments that are generated purely as a result of acceptable practice variability, or subjective judgements which cannot be modelled in a computer system. This filtering should increase the acceptability of the system by reducing the total number of comments produced, while increasing the average importance of those comments that remain.

In order to evaluate the significance of an individual error, it is necessary to develop a more general understanding of *error types* in the trauma domain, and how the occurrence these different types of errors affects the quality of care. In the absence of an objective "gold standard" for evaluating trauma management plans, I plan to approximate an expert's assessment of the significance of errors in my system, using data from the evaluations of expert judges on a set of actual case descriptions as a guide.

For the purposes of this project, we will partition errors into three equivalence classes according to their significance:

1. Tolerable, probably harmless.

2. Non-critical, but potentially harmful.

3. Critical, potentially fatal.

Which of these classes a particular error belongs to will determine how it will be handled by the critique. Errors in the first class will not be mentioned at all, errors in the second class will appear as simple reminders or statements, while errors in the third class will appear as warnings.[4]

In order to partition identifiable errors into these three classes, it is necessary to identify sets of features by which to characterize them. The potential costs of an error cannot be quantified solely in terms of time or money. We must also consider

[4]The system could be enhanced by allowing the physician to select a level of "pickyness" in which case whether or not errors in the first two classes would be mentioned would be dependent on the preferences of the user.

such issues as potential risk, the patient's discomfort or pain, and conformance to standard practice guidelines.

To categorize the errors that TraumaTIQ is able to recognize in terms of their significance, the following features will be considered:

1. The type of error: omission, commission, procedure choice, or temporal order.
2. The type of goals involved: diagnostic or therapeutic.
3. The type of diagnosis being addressed: primary or secondary.
4. Whether actions are invasive or non-invasive.
5. The urgency of actions.
6. The cost and risk of actions (represented numerically).
7. The priority of actions according to standard medical priorities of airway, breathing, circulation, etc.
8. The type of diagnostic action: bedside investigation, monitoring, or diagnostic procedure.
9. The type of therapeutic action: stabilizing, definitive, or adjuvant therapy.
10. For errors of commission, the reason the action is unmotivated.
11. For procedure choice errors, the reason for choosing procedure α rather than procedure β to address the goal Γ.

In order to develop criteria for assigning particular errors to one of the three categories above, I intend to make use of the data from a retrospective evaluation of TraumAID's performance as a management planner [9, 41]. These data consist of a step-by-step evaluation of the management plans produced by two versions of the TraumAID system (TraumAID 1.0 and TraumAID 2.0), as well as the actual care provided, for 97 real trauma cases. Three trauma surgeons evaluated all three versions of each case. In addition, six national trauma experts evaluated the 97 management plans produced by TraumAID 2.0. In addition to an overall assessment of the quality of care, these evaluations include judgments about errors of commission, errors of omission, and scheduling errors for individual actions in the plans. Errors that led to assigning the plan an unacceptable or nearly unacceptable rating were marked as such.

A preliminary look at the judgment data shows that the judges rarely agree on the level of individual errors. However, it appears that the use of higher-level features to characterize errors will yield some predictive power. For example, the

omission of standard workup procedures, even when not necessitated by the patients particular injury, tends to be marked as a significant error.

Using the data from this evaluation, I will attempt to find correlations between the occurrence of particular features, such as those listed above, in the management plans and particular comments from the judges. I will use this information to develop criteria to decide whether an error should be classified as *tolerable*, *non-critical*, or *critical*. Further evidence for these criteria will come from the observation of naturally occurring critiques on videotapes of trauma management sessions, and from discussions with trauma experts. This evidence will serve as a basis for deciding whether, and in what manner, an item identified as an error by TraumaTIQ should be included in the critique presented to the physician.

4.5 Critique Generation

The output of the plan evaluation stage represents the *communicative goals* of the critique, i.e. what information will be conveyed to the physician. The next stage is to realize those goals via the generation of linguistic output. This stage of the critiquing process is not yet fully implemented in the TraumaTIQ system, but I will outline here how it will be done. In keeping with the approach seen in the language generation literature [33], the generation process in TraumaTIQ will be separated into two stages: strategic (deep) generation, which involves determining the content and structure of the output, and tactical (surface) generation, in which the actual words and phrases are chosen and put together to produce written or spoken natural language. For the purposes of this project, I will be concerned primarily with the issues associated with strategic generation – determining exactly what to say, how to represent concepts for the purposes of language generation, and how to organize the output. The problem of tactical generation will be addressed elsewhere (see [37] for an approach to tactical generation applied to the trauma domain.)

4.5.1 Strategic Generation

Determining critique content

The contents of the critique in this system are driven by the output of the plan evaluation routines, which are responsible for identifying information that should be reported to the physician. Corresponding to the four types of errors recognized during plan evaluation, the propositions to be conveyed to the physician will concern errors of omission, errors of commission, procedure choice errors, and scheduling errors. Depending on the level of significance of the particular error, the comment will be assigned an illocutionary force of either INFORM or WARN.[5] The illocu-

[5] I do not adopt the strategy taken by Rankin [38] of commenting on every action proposed by the physician. While this strategy has the advantage of convincing the physician that the system

tionary force of a comment could ultimately influence the phrasing and, in the case of spoken critiques, the intonation of the output.

In addition to informing the physician of potential errors in his plan, justifications will be included in support of important points. However, the level of explanation currently available by directly accessing TraumAID's knowledge-base is limited to the information needed by the system for its planning and reasoning. Since TraumAID's knowledge is encoded in rules that tend to gloss over the details of biomedical knowledge underlying them, explanations derived from these rules will not contain such details. On the other hand, since the system is designed to be used by trained physicians who presumably already have a background in this area, detailed explanations may not be necessary, or even desirable. For example, consider the following possible critique:

> "A chest tube should be inserted to treat the massive hemothorax before getting the X-ray of the abdomen. This is because of the urgency of treating the hemothorax."

This comment presupposes that the physician already knows that a massive hemothorax must be attended to urgently, but suggests that he may have overlooked it for some reason. Further explanation as to *why* the goal of treating a massive hemothorax is urgent is not currently available in the TraumAID system. Future additions to the system could include a more comprehensive explanation facility, so that the system would be able to present more basic-level explanation and discussion if desired. This explanation capability could be interactive, so that the physician could request additional justification or clarification regarding elements of the critique.

In an on-line critique, it is necessary to take into account what has already been said to the physician. Therefore, the system keeps a record of the comments it has already produced, and assumes that the physician is aware of the information they contain. The question of whether, or how often to repeat comments is an open one. If the physician continues with his current course of action in spite of a critique, it is probably necessary to repeat the comment since he may not have heard it or payed attention to it the first time. However, it may be the case that the physician has heard the critique and has simply chosen to ignore it. In such a case, it would be undesirable for the system to keep repeating its comment. This issue will be resolved through on-site experimentation with the system.

Structure of the critique output

In terms of the organization of the critique, TraumaTIQ will conform with standard medical practice and organize its comments with respect to the goals that they address, with the goals listed in order of priority. In this way, multiple comments

---

has considered all of his proposals, I do not believe that it is appropriate or necessary in a crisis management situation to confirm each undisputed action.

related by a single goal will be output sequentially. Since the critiques produced by this system are to be delivered during a time-critical management session, they will not be constructed as multi-sentence texts. I will therefore not be concerned in this project with issues such as the rhetorical relations necessary for producing coherent critiquing prose.

4.5.2 Concept Representation in TraumAID

Currently, TraumaTIQ displays its critiques as single-sentence comments produced by inserting specific action and goal names into stored templates. This often results in awkward-sounding output because the concept names that TraumAID uses in its reasoning are not always appropriate slot fillers. For example, goals are referred to using strings beginning with either RO (Rule Out) for diagnostic goals, or Rx for therapeutic goals. Without any semantic decomposition of these strings, we end up with output like:

> "Please consider performing a urinalysis to address the goal RO hematuria."

Even worse, the action names are either names of tests or procedures or descriptions of actions. Using the same template as the above example, we might produce the string,

> "Please consider performing a close chest wound to address the goal Rx open sucking chest wound."

where Close_Chest_Wound is the name of an action.

These sentences could be made less awkward in an ad hoc manner by assigning an English translation to each concept, such as "closing the chest wound" for Close_Chest_Wound and "performing a urinalysis" for urinalysis. However, this would not be a satisfactory solution since it would not allow concepts to appear easily in different sentence positions. For example, suppose we wanted the system to produce reminders such as,

> "Reminder: You have indicated that you intend to close the chest wound, but you have not yet done it after X minutes."

This would require a different realization of the concept Close_Chest_Wound.

The underlying problem is that the representation of concepts in the system's knowledge base was designed for the purposes of reasoning and planning, not for generating English sentences. To improve the quality of the output, a more general semantic decomposition of these concepts must be available, representing the relationships between the main action, its recipient, and their various modifiers. This representation, together with an appropriate grammar and lexicon, can then be used to generate sentences conveying any number of propositions that we may

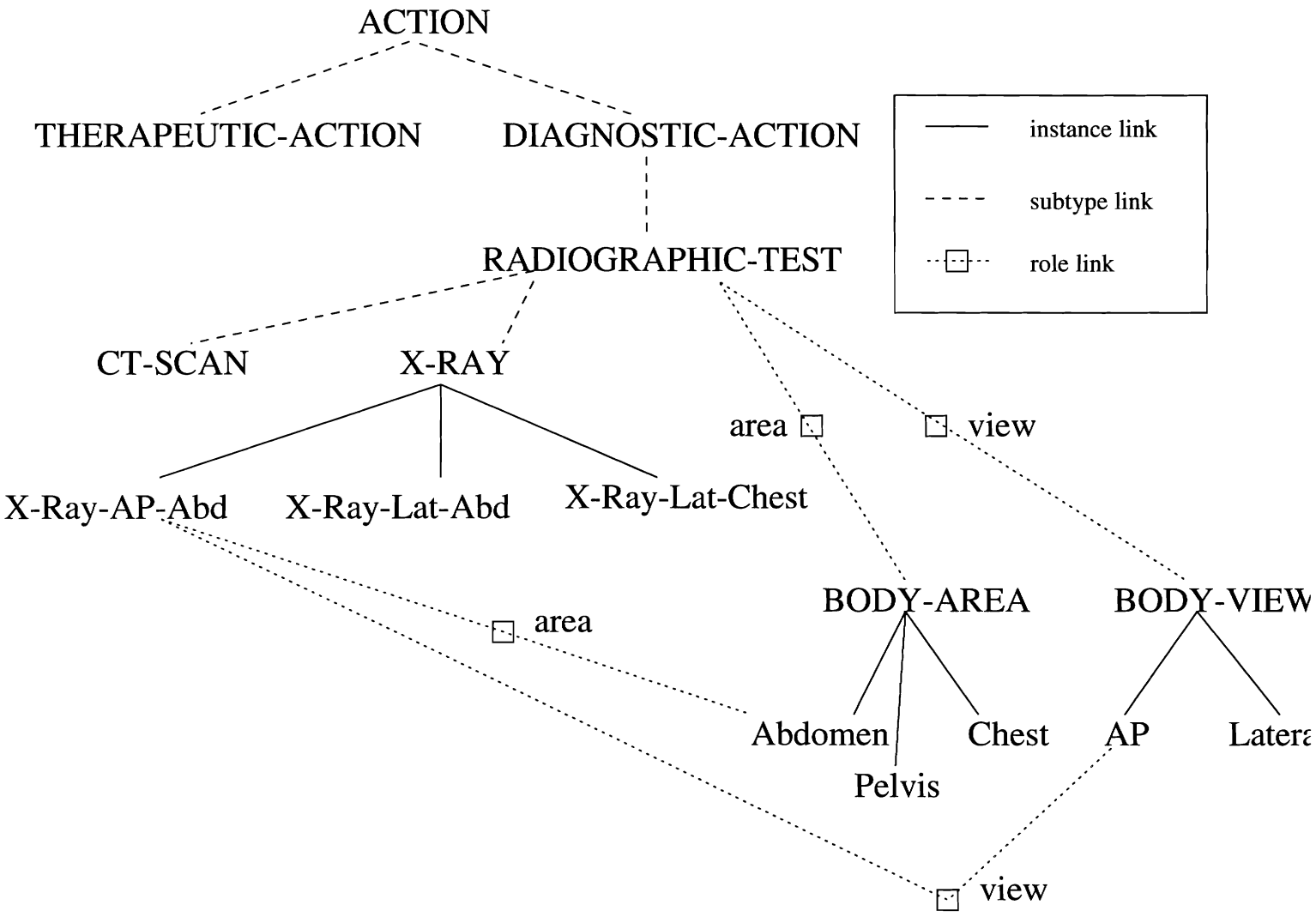


Figure 4.3: The action hierarchy for X-rays

decide should be included in the critique. Figure 4.3 shows a portion of the action hierarchy that I have begun to develop to address this problem.

This concept decomposition has the important property that it makes it possible to identify *contrast elements* within a single comment. An important function of the critique is to suggest alternatives to proposed actions. These contrasted actions can be quite similar, such as an AP abdominal X-ray[6] compared to a lateral abdominal X-ray. The ability to pick out the point of contrast between these two actions (in this case AP vs. lateral) will allow us to stress that contrast, either with larger or bolder text in a written critique, or, as I will discuss further in the following section, with prosodic stress in a spoken critique.

4.5.3 Tactical Generation

The tactical generation of language from a semantic representation of propositional content is an important area of research in its own right, and one that I do not propose to solve in this dissertation. I will include here a brief discussion of an approach that could be used in the future to generate comments in both written text and natural-sounding synthesized speech.

In [36], Prevost and Steedman describe their "functional head-driven, top-down approach" to tactical generation using a Combinatory Categorial Grammar (CCG) formalism. In their paper they discuss how this approach can be used to generate situationally appropriate prosodic stress contours in spoken language output. Given a semantic representation of the output, in which the *theme* (what the proposition is about) and *rheme* (what new information the proposition has to say about the theme) are marked, they produce a surface string that is marked with pitch accents appropriate to the information content *and* the contextual meaning of the proposition. This technique can dramatically increase the hearer's ability to grasp the meaning of an utterance, particularly in a situation where a contrast is being made. For example, using a default lexical stress pattern for the word "thoracotomy," with the primary lexical stress on the third syllable, would produce the following spoken output:

> "A left thoraCOTomy is more appropriate than a right thoraCOTomy for this patient."

Where the contextually correct intonation would be:

> "A LEFT thoracotomy is more appropriate than a RIGHT thoracotomy for this patient."

The latter would be much easier for a listener to interpret and ascribe the correct meaning to because it emphasizes the contrast between the two elements being

[6] AP stands for anterior-posterior, i.e. the view from front to back.

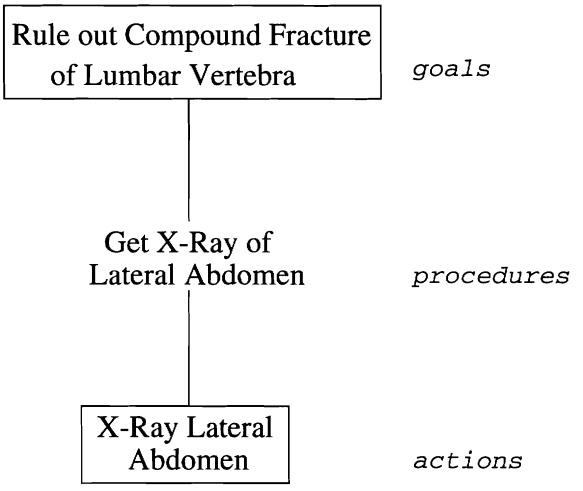


Figure 4.4: Example 1: The physician's plan after ordering X-ray

compared. However, the former intonation would be more appropriate in some cases, such as in the sentence:

> "A left thoraCOTomy is more appropriate than a left thoraCOSTomy for this patient."

The fact that different intonational contours are appropriate depending on the context of the sentence shows that *no* default algorithm will work for every utterance. Rather, a procedure that takes account of the semantics of the utterance is needed.

4.6 Four Examples

As an example of the critiquing process in TraumaTIQ, suppose we have a patient with a gunshot wound to the abdomen and loss of sensation in both legs. These findings lead TraumAID to activate the goal of diagnosing a fracture of the lumbar vertebrae (RO-COMPOUND-FRACTURE-LUMBAR-VERTEBRA). TraumAID knows two procedures that can address this goal, a lateral abdominal X-ray (GET-X-RAY-LAT-ABD), or an abdominal CT-scan (GET-CT-SCAN-ABD). The former is preferred as it takes less time.

Figure 4.4 shows what happens if the physician orders a lateral abdominal X-ray in this situation. Since this action is currently in TraumAID's plan, TraumaTIQ's plan inference routines will ascribe the associated goal, diagnosing a compound fracture of the lumbar vertebrae, to the physician (together with any other goals that it believes are relevant and can be addressed by the same procedure). It will also ascribe the intention to perform an abdominal X-ray in order to address that goal. Since there is no discrepancy between this inferred plan and the plan

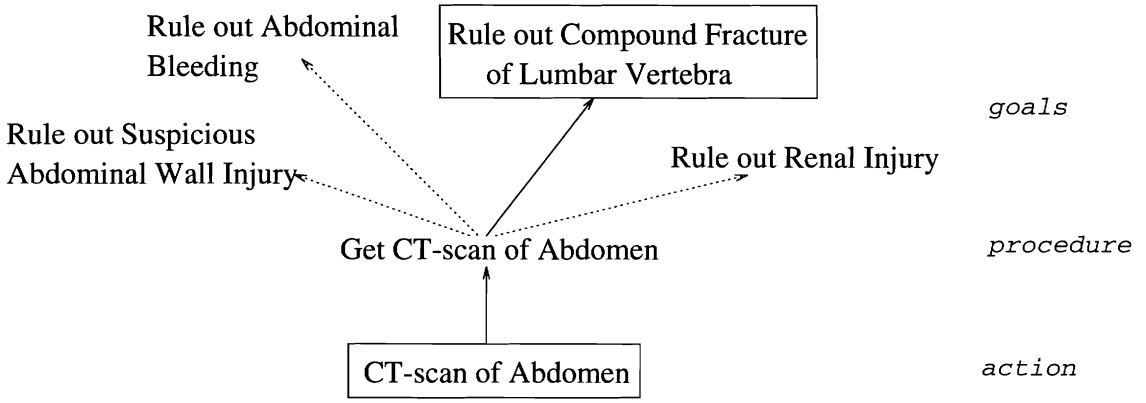


Figure 4.5: Example 2: The physician's plan after ordering CT-scan

produced by TraumAID, plan evaluation will result in no communicative goals, and no critique will be produced.

Figure 4.5 shows the situation where the physician has ordered an abdominal CT-scan instead of the recommended X-ray, and the system can see no better reason for doing the CT-scan. Since one of the goals that might motivate this action, diagnosing a compound fracture of the lumbar vertebrae, is currently on TraumAID's list of relevant goals to be addressed, TraumaTIQ will infer that the physician intends to do a CT-scan in order to address that goal. Now, since there *is* a discrepancy between the way TraumAID has chosen to address the goal and the way the system believes that the physician intends to address it, assuming that this is seen as a significant difference the plan evaluation phase will result in the communicative goal of suggesting that a lateral abdominal x-ray be done instead of an abdominal CT-scan to diagnose a compound fracture of the lumbar vertebrae. This goal will then be translated into the sentence we have seen previously:

> "Consider getting a lateral X-Ray of the abdomen rather than getting a CT-scan of the abdomen, to check for fracture of the lumbar vertebrae."

In the third example the patient shows hematuria (blood in the urine) as well as the previous findings of abdominal gunshot wound and loss of sensation in both legs. Hematuria leads to a goal of diagnosing renal injury. The only procedure TraumAID knows for diagnosing renal injury is an abdominal CT-scan. Rather than including both a CT-scan *and* an X-ray in its management plan, TraumAID optimizes the plan so that both goals (diagnosing renal injury and diagnosing fractured vertebrae) are covered by the CT-scan procedure. If the physician were now to order a CT-scan, TraumaTIQ would recognize this action as being motivated by two currently active goals, and ascribe *both* of them to the physician, as in Figure 4.6. Once again, since there is no discrepancy between TraumAID's plan and the inferred physician's plan, no critique will be produced.

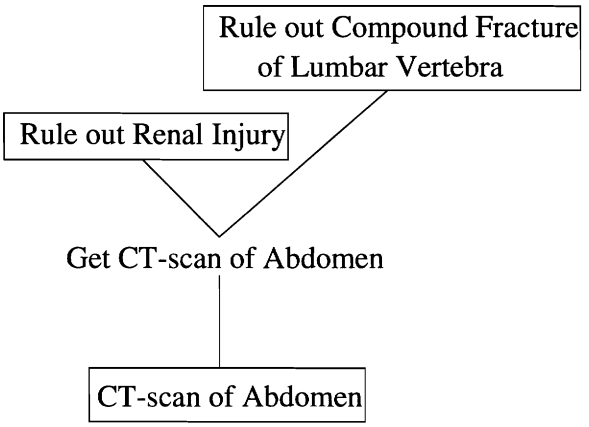


Figure 4.6: Example 3: The physician's plan after ordering CT-scan with finding of hematuria

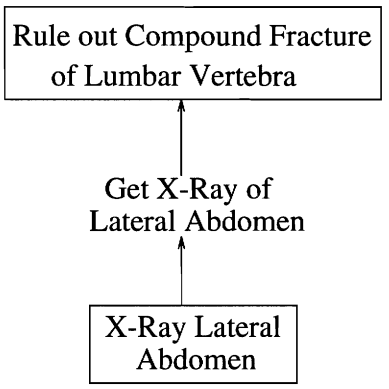


Figure 4.7: Example 3: The physician's plan after ordering X-ray with finding of hematuria

The final example (Figure 4.7) shows what would happen in the same situation as Example 3, if a lateral abdominal X-ray alone were ordered, with no CT-scan. TraumaTIQ infers that the X-ray has been ordered to address the active goal of diagnosing a fracture of the lumbar vertebrae. However, it also finds that the goal of diagnosing renal injury has not been addressed in the physician's plan. This situation results in two comments from TraumaTIQ:[7]

> "Consider getting an abdominal CT-scan now to check for renal injury."
>
> "Consider getting a CT-scan of the abdomen rather than getting a lateral X-Ray of the abdomen to check for fracture of the lumbar vertebrae.

[7]The first sentence will only be produced after a reasonable amount of time has been allowed for the physician to order a CT-scan. The second sentence is slightly different from the PROCEDURE-CHOICE template seen earlier, since it includes the reason for preferring a CT-scan in this situation. In the current TraumAID knowledge base, the reasons for procedure preferences are not always available.

A CT-scan of the abdomen can also be used to check for renal injury."

4.6.1 Analysis of naturally occurring critiques

In considering how to design a critiquing system that produces critiques with appropriate content and phrasing, the question of how people actually critique each other arises. In order to explore this question, I plan to observe and analyze videotapes of Trauma Alert sessions from the Medical College of Pennsylvania. The aim is to get examples of naturally occurring critiques in the domain of trauma management, to serve as a basis for the critiques to be delivered by the system. I will be looking at both the content and the surface form of these dialogues. In terms of the content I am interested in what kinds of comments are offered by physicians observing their colleagues. Do they mostly critique technique, ie. *how* a procedure is carried out? Do they point out inefficiencies in a basically adequate plan? Do they question underlying goals if an action does not seem relevant? In terms of form, I am interested in the type of phrasing the physicians use to convey their critiques, particularly as it changes with the relative levels of experience of the participants.

In addition to analyzing the content and form of the critiques, I plan to do a retrospective evaluation of the videotape data by discussing them with trauma surgeons (the ones that were involved in the particular cases and others), and getting their opinions as to the correctness and effectiveness of the comments. This will serve as a first approximation of how my system should behave in order to be effective.

4.7 Evaluating the System

In order to evaluate the critiquing system fully, it would be necessary to demonstrate a positive effect on the occurrence of errors in actual care provided while the system is in use. While this extensive evaluation is outside of the scope of my dissertation work, I do intend to evaluate the system in the following two ways:

1. An informal investigation of physicians' reactions to the two different modes of decision support: presentation of a recommended plan vs. critiquing. A prospective evaluation of TraumAID 2.0 is scheduled to begin shortly, with the purpose of examining when and how the system can influence physician's behavior. For the purposes of this evaluation, the system's recommendations will be presented to the physician in graphical form, in a manner designed to make them as easy to interpret as possible. Actions will be linked to their intended goals, important goals will be more prominent in the presentation, and actions that are contingent on the outcome of other actions will not be

presented. In order to evaluate the potential advantages (or disadvantages[8]) of employing a critiquing approach for the TraumAID system, I will compare the physicians' reactions to this graphical mode of output delivery to their reactions to the critiquing mode.

2. A comparison of the critiques produced by the system with critiques produced by actual trauma surgeons. It has been shown [9, 48] that there is often little agreement between physicians on what constitutes an error that should be commented on. However, I will attempt to show that the performance of the TraumaTIQ system is at least comparable, in terms of the number and types of comments, to the critique that would be produced by a physician observing the same management plan.[9] I will further evaluate the computer-generated critiques by showing transcripts of cases, together with their critiques, to a panel of experts, and asking them to judge the relevance and significance of the comments produced.

[8]It is possible that the limited information about the system's reasoning processes available in a critique could be seen as a drawback.

[9]The computer-generated critique has the advantage that it is guaranteed to be consistent, unlike a human critic.

Chapter 5

Conclusions and Future Work

In this proposal, I have presented an argument for the claim that, in domains characterized by (1) multiple interacting goals, (2) time-critical decision making, and (3) task-centered activity, human-computer interaction based on a propose-and-critique model is preferable to the traditional expert-system approach. The critiquing approach has advantages in terms of psychological acceptability, as well as flexibility in handling variations in practice and subjective judgments.

Furthermore, I have shown how a combination of integrated knowledge structures and reasoning capabilities can be used to produce and update critiques in real time, *during* the construction and execution of a management plan. My proposed critiquing architecture comprises:

1. A plan inference component that uses knowledge about actions and goals in the domain, together with knowledge about the specific situation, to infer a model of the user's goals and intentions from his proposed actions.

2. A plan evaluation component that makes use of knowledge about policy, practice guidelines, etc. and how they should shape behavior in a given situation, in order to identify errors that will then be mentioned in the critique.

3. A critique generation component that converts the results of plan evaluation into a concise and coherent natural language critique.

In the first phase of this research, I have developed and implemented a basic architecture for a real-time critiquing system that can detect and respond to a range of planning failures. Currently, the TraumaTIQ system is capable of determining when the physician's plan does not correspond to the standards set by TraumAID, and of responding appropriately and in a timely manner. Clearly, however, there is a great deal of work yet to be done:

1. Enhancing the knowledge base with more explicit information about the system's reasoning, to provide for better explanations. In particular, local pref-

erences for one procedure over another to address a goal are not currently justified in the knowledge base.

2. Developing criteria for assessing the significance of error types, so that insignificant discrepancies between TraumAID's recommendations and the physician's proposal will not be critiqued.

3. Looking at naturally occurring critiques on videotape, and talking to practicing trauma surgeons, so as to identify the best way to phrase computer-generated critiques.

4. Developing a better means of sentence generation to replace the current use of canned text, so that the phrasing of comments more clearly reflects their content.

5. Evaluating the system in terms of the relevance and correctness of the critiques it produces.

In closing, I would like to mention a few related issues which I believe would be interesting areas of research in their own right. First, I have sidestepped the question of user modelling in my system, on the assumption that all of the intended users will have comparable expertise and knowledge and thus modelling individual users is not necessary. Whether or not this assumption is valid, the question of what a user model might add to the system in terms of tailoring the critique to a particular user remains an open one. For example, the system could maintain a database of physicians that it regularly "works with" along with the types of errors they tend to commit, specific information about their expertise with various procedures, or their particular preferences for handling certain problems. This information could be used to provide a critique that is more relevant to the individual user.

A second issue is the applicability of the critiquing architecture for use as a educational tool. Trainees in the area of trauma care could use the TraumAID system to run through simulations of cases, receiving feedback from the critiquing module regarding the quality of their management decisions. In order to be appropriate for this type of application, the critique would have to assume a quite different stereotype of user, and thus would have to include more basic explanations. It also might be the case that certain assumptions the system currently makes about the correctness of the user's goals would not be valid, in which case the plan recognition component of the system would have to be significantly altered.

The critiquing approach encompasses a wide variety of domains and implementations. What they have in common is the recognition that the traditional role of expert systems as decision-making advisors must be questioned and reinterpreted if we are to benefit as much as possible from the power of knowledge-based systems. The investigation of critiquing can lead to a better understanding of how computers can interact effectively with humans to influence their behavior, whether in urgent, time-critical situations or under less strenuous circumstances.

5.1 Acknowledgements

Many thanks to my advisor Bonnie Webber for her help with this proposal and the ideas that went into it. I would also like to thank John Clarke, Sandra Carberry, Ron Rymon, Jonathan Kaye, and the rest of the TraumAID group. This work has been supported by the National Library of Medicine under grant R01 LM05217-01.

Bibliography

[1] James F. Allen and C. Raymond Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.

[2] J.G. Anderson, S.J. Jay, H.M. Schweer, and M.M. Anderson. Why doctors don't use computers: some empirical findings. *Journal of the Royal Society of Medicine*, 79:142–144, March 1986.

[3] J.A. Bateman and C.L. Paris. Phrasing a text in terms the user can understand. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[4] Karen E. Bradshaw, Reed M. Gardner, and T. Allan Pryor. Development of a computerized laboratory alerting system. *Computers and Biomedical Research*, 22:575–587, 1989.

[5] Sandra Carberry. Modeling the user's plans and goals. *Computational Linguistics*, 14(3):23–37, 1988.

[6] Eugene Charniak and Robert P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

[7] Brant Cheikes. *Planning Responses From High-Level Goals: Adopting the Respondent's Perspective in Cooperative Response Generation.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1992.

[8] J. R. Clarke. Clinical surgical decision making. In I.M. Rutkow, editor, *Socioeconomics of Surgery*, chapter 19, pages 315–331. Mosby, St. Louis, 1989.

[9] J. R. Clarke, R. Rymon, B. L. Webber, C. Hayward, T. Santora, D. Wagner, C. Friedman, A. Ruffin, C. Blank, L. Nieman, and A. B. Eastman. Computer-generated protocols: A tool for quality control during the initial definitive management of trauma patients. Submitted to the Annual Meeting of the *American Assoc. for the Surgery of Trauma*, September 1993.

[10] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[11] G. Fischer, A. C. Lemke, T. Mastaglio, and A. I. Morch. The role of critiquing in cooperative problem solving. *ACM Transactions on Information Systems*, 1991.

[12] Gerhard Fischer, Andreas C. Lemke, and Thomas Mastaglio. Critics: An emerging approach to knowledge-based human computer interaction. In *Conference on Human Factors in Computing Systems*, 1990.

[13] Gerhard Fischer, Kumiyo Nakakoji, and Jonathan Ostwald. Critics: Facilitating knowledge delivery and knowledge construction in integrated design environments. In *AAAI-93 Workshop on Expert Critiquing Systems*, 1993.

[14] Bradley A. Goodman and Diane J. Litman. On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction*, 2(1-2):55–82, 1992.

[15] Barbara J. Grosz and Candace L. Sidner. Attentions, intentions and the structure of discourse. *Computational Linguistics*, 12:175–204, 1986.

[16] Karen E. Huff and Victor R. Lesser. Integrating plausible reasoning in an incremental plan recognizer. Technical Report 93-72, University of Massachusetts, Amherst, 1993.

[17] Daniel Kahneman, Paul Slovic, and Amos Tversky, editors. *Judgement under uncertainty: Heuristics and biases.* Cambridge University Press, Cambridge, 1982.

[18] Henry Kautz. A circumscriptive theory of plan recognition. In Jerry Morgan Philip R. Cohen and Martha E. Pollack, editors, *Intentions in Communication.* Bradford Books, 1990.

[19] J.L. Kolodner, E.A. Domeshek, and C.M. Zimring. Case-based design critiquing. In *AAAI-93 Workshop on Expert Critiquing Systems*, pages 109–114, 1993.

[20] Lynne Lambert and Sandra Carberry. A tripartite, plan-based model of dialogue. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, 1991.

[21] C. P. Langlotz and E. H. Shortliffe. Adapting a consultation system to critique user plans. *International Journal of Man-Machine Studies*, 19:479–496, 1983.

[22] Diane Litman and James Allen. Discourse processing and commonsense plans. In Jerry Morgan Philip Cohen and James Allen, editors, *Intentions in Communication.* MIT Press, 1989.

[23] Robert London. Student modeling to support multiple instructional approaches. *User Modeling and User-Adapted Interaction*, 2(1-2):117–154, 1992.

[24] Robert London and William J. Clancey. Plan recognition strategies in student modelling: prediction and description. In *Proceedings of the American Association for Artificial Intelligence*, pages 335–338, Pittsburgh, 1982.

[25] W.C. Mann and S.A. Thomson. Rhetorical structure theory: Description and construction of text structures. In Gerard Kempen, editor, *Natural Language Generation*, pages 83–96. Martinus Nijhoff, 1987.

[26] Clement J. McDonald, Siu L. Hui, David M. Smith, William M. Tierney, Stuart J. Cohen, Morris Weinberger, and George P. McCabe. Reminders to physicians from an introspective computer medical record. *Annals of Internal Medicine*, 100:130–138, 1984.

[27] P. L. Miller. *A Critiquing Approach to Expert Computer Advice: ATTENDING.* London: Pittman Press, 1984.

[28] P. L. Miller. *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer.* New York: Springer-Verlag, 1986.

[29] V.O. Mittal and C.L. Paris. Text generation: Explanation vs. criticism in expert systems. In *AAAI-93 Workshop on Expert Critiquing Systems*, 1993.

[30] J. D. Moore and W.R. Swartout. A reactive approach to explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[31] R. Neches, W. R. Swartout, and J. Moore. Explainable and maintainable expert systems. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 382–389, Los Angeles, 1985.

[32] Cécile Paris. Generation and explanation: Building an explanation facility for the explainable expert systems framework. In C. Paris, W. Swartout, and W. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 49–81. Kluwer Academic Publishers, 1991.

[33] Cecile L. Paris, William R. Swartout, and William C. Mann, editors. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers, Boston, 1991.

[34] Martha E. Pollack. *Inferring domain plans in question-answering.* PhD thesis, University of Pennsylvania, 1986.

[35] Martha E. Pollack. Plans as complex mental attitudes. In *Cohen, Morgan and Pollack, eds. Intentions in Communication, MIT Press, Cambridge, MA.*, 1990.

[36] Scott Prevost and Mark Steedman. Generating contextually appropriate intonation. *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, 1993.

[37] Scott Prevost and Mark Steedman. Using context to specify intonation in speech synthesis. In *Proc. EuroSpeech '93*, Berlin, Germany, 1993.

[38] I. Rankin. The deep generation of text in expert critiquing systems. Licentitate Thesis no.184, Linköping University, 1989.

[39] D. Rochowiak, J. Rogers, and S. Messimer. Composite design and manufacturing critiquing system. In *AAAI-93 Workshop on Expert Critiquing Systems*, 1993.

[40] R. Rymon, B. Webber, and J. Clarke. Progressive horizon planning: Planning exploratory-corrective behavior. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6), November 1993. Special Issue on Planning, Scheduling and Control.

[41] Ron Rymon. *Diagnostic Reasoning and Planning in Exploratory-Corrective Domains (Appears as Technical Report MS-CIS-93-84).* PhD thesis, University of Pennsylvania, 1993.

[42] Candace Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1, 1985.

[43] Barry G. Silverman. Expert critics: operationalizing the judgement/ decisionmaking literature as a theory of "bugs" and repair strategies. *Knowledge Acquisition*, 3:175–214, 1991.

[44] Barry G. Silverman. Survey of expert critiquing systems: Practical and theoretical frontiers. *Communications of the ACM*, 35(4):106–127, 1992.

[45] Barry G. Silverman. Expert bias research: Issues confronting knowlege engineers. In *AAAI-93 Workshop on Expert Critiquing Systems*, 1993.

[46] R. L. Teach and E. H. Shortliffe. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Computers and Biomedical Research*, 14:542–558, 1981.

[47] Amos Tversky and Daniel Kahneman. Judgement under uncertainty: heuristics and biases. In Daniel Kahneman, Paul Slovic, and Amos Tversky, editors,

Judgement under uncertainty: heuristics and biases, chapter 1, pages 3–20. Cambridge University Press, Cambridge, 1982.

[48] J. van der Lei. *Critiquing Based on Computer-Stored Medical Records.* PhD thesis, Erasmus University, 1991.

[49] B. L. Webber, R. Rymon, and J. R. Clarke. Flexible support for trauma management through goal-directed reasoning and planning. *Artificial Intelligence in Medicine*, 4, 1992.