March 1996

Critiquing: Effective Decision Support In Time-Critical Domains

Abigail S. Gertner
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS 96-03.

Critiquing: Effective Decision Support In Time-Critical Domains

Abstract

The TraumAID system is a tool for assisting physicians during the initial definitive management phase of patients with severe injuries. Originally, TraumAID was conceived as a rule-based expert system combined with a planner. After this architecture had been implemented and evaluated, we began to face the issue of how TraumAID could communicate its plans to physicians in order to influence their behavior and have a positive effect on patient outcome. It was hypothesized that a *critiquing* approach, in which the system is told what actions the user intends to carry out and produces a critique in response to those intentions, might be appropriate.

To meet the needs of physicians engaged in managing trauma cases, critiques must be updated and made available rapidly. They must be clear and succinct, containing only relevant information while still including enough justification to make them convincing. To address the need for these features in the system, I have developed a critiquing architecture consisting of three components, incremental plan recognition, plan evaluation, and critique generation. I have implemented this architecture in TraumaTIQ, a critiquing interface for the TraumAID system. Comparison of TraumaTIQs comments on 97 actual management plans with comments made by three local trauma experts showed that TraumaTIQ produced 48.3% of comments made by only one judge, and 70.27% of comments made by two or more judges. This relationship between inter-judge agreement and judge-system agreement is statistically significant. In addition, regression analysis shows that TraumaTIQs plan evaluator is a significant predictor of the judges' overall case ratings.

This approach to communicating with physicians in time critical domains has the advantage that it is user-focused, minimally intrusive, and quick to respond to potential errors even on the basis of partial information. Unlike previously developed reminder and alert systems, TraumaTIQ evaluates the physician's proposed plan and attempts to intervene before problems occur. And unlike previous critiquing systems, it is able to provide ongoing decision support during the planning and delivery of care. In the context of time-critical patient management it is, therefore, a more appropriate form of interaction.

Comments

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS 96-03.

Institute for Research in Cognitive Science

Critiquing: Effective Decision Support In Time-Critical Domains

(Ph.D. Dissertation)

Abigail S. Gertner

University of Pennsylvania
3401 Walnut Street, Suite 400C
Philadelphia, PA 19104-6228

March 1996

Site of the NSF Science and Technology Center for Research in Cognitive Science

IRCS Report 96--03

CRITIQUING: EFFECTIVE DECISION SUPPORT IN TIME-CRITICAL DOMAINS

ABIGAIL S. GERTNER

A DISSERTATION

in

COMPUTER AND INFORMATION SCIENCE

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy.

1995

Bonnie L. Webber

Supervisor of Dissertation

Peter Buneman

Graduate Group Chairperson

Acknowledgments

This dissertation owes its existence in large part to the support and encouragement of my advisor Bonnie Webber and to John Clarke, from whom I have learned a great deal about many important things including AI, the management of trauma patients, and surviving life in academia. When I think about how I have metamorphosed in the past five years from an innocent psychology major into a "real" computer scientist, I realize how much I have benefited from their guidance.

I would also like to thank the other members of my committee: Sandra Carberry, Mark Steedman and Aravind Joshi, for their advice and insight in helping me pull this project together.

Special thanks go to the members of the TraumAID group, with whom I have shared many early Tuesday mornings over the years. The group has grown and changed as the project has progressed, and I will not list all the past and present members here – you know who you are! But I would especially like to thank Jonathan Kaye for his technical expertise, and his inability to finish a sentence, and Ron Rymon, whose work on TraumAID provided a jumping-off point for my own research.

When it came time to evaluate TraumaTIQ's performance, I received invaluable assistance from Guy Whitten. Guy deserves a special note of thanks for helping me with the statistical analysis under difficult circumstances.

On the personal side, the list of people who have made my grad school experience fun and kept me more or less sane through it all is too long to enumerate here. I definitely have to mention my cat, Cosmo, and his friends Moose and Milo. No one should have to write a dissertation without cats. I would also like to thank my housemates, past and present, especially Mike White and Mark Little for producing a never-ending supply of homebrew.

Thanks also to Jeff Achter for teaching me to rock climb – another great distraction.

The past year has been a difficult and stressful one for me and I owe a great deal to Scott Pauls who has been incredibly supportive and not *too* distracting through it all!

Last but not least I thank my brothers for giving me something to compete with, and my parents, without whom I would certainly not be here.

Abstract

Critiquing: Effective Decision Support in Time-Critical Domains

Abigail S. Gertner

Bonnie L. Webber (Supervisor)

The TraumAID system is a tool for assisting physicians during the initial definitive management phase of patients with severe injuries. Originally, TraumAID was conceived as a rule-based expert system combined with a planner. After this architecture had been implemented and evaluated, we began to face the issue of how TraumAID could communicate its plans to physicians in order to influence their behavior and have a positive effect on patient outcome. It was hypothesized that a *critiquing* approach, in which the system is told what actions the user intends to carry out and produces a critique in response to those intentions, might be appropriate.

To meet the needs of physicians engaged in managing trauma cases, critiques must be updated and made available rapidly. They must be clear and succinct, containing only relevant information while still including enough justification to make them convincing. To address the need for these features in the system, I have developed a critiquing architecture consisting of three components: incremental plan recognition, plan evaluation, and critique generation. I have implemented this architecture in TraumaTIQ, a critiquing interface for the TraumAID system. Comparison of TraumaTIQ's comments on 97 actual management plans with comments made by three local trauma experts showed that TraumaTIQ produced 48.3% of comments made by only one judge, and 70.27% of comments made by two

or more judges. This relationship between inter-judge agreement and judge-system agreement is statistically significant. In addition, regression analysis shows that TraumaTIQ's plan evaluator is a significant predictor of the judges' overall case ratings.

This approach to communicating with physicians in time-critical domains has the advantage that it is user-focused, minimally intrusive, and quick to respond to potential errors even on the basis of partial information. Unlike previously developed reminder and alert systems, TraumaTIQ evaluates the physician's proposed plan and attempts to intervene *before* problems occur. And unlike previous critiquing systems, it is able to provide ongoing decision support *during* the planning and delivery of care. In the context of time-critical patient management it is, therefore, a more appropriate form of interaction.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The scenario

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. She 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? She must have a model 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. She also must understand the problem, and the approach he considers best for addressing it, in order to be able to recognize errors and offer alternatives to the faulty plan. She should be able to explain her own reasoning so as to justify her advice to the resident. In addition, the advisor should have some idea of how to communicate cooperatively in order to influence the resident's future actions.

This dissertation describes an approach to providing real-time decision support in complex, task-oriented situations that is based on this model of advice giving. As the above

example suggests, this problem has been addressed in the context of the management of multiple trauma. This task typically involves reasoning about multiple goals, integrating diagnosis and treatment into a single management plan, efficient allocation of resources, and acting under time pressure. In such situations, we recognize both the need for intelligent decision support, and the obstacles to providing such support arising from both the heavy cognitive demands of the task, and the reluctance of certain kinds of users to accept advice from computers. We have attempted to minimize obstacles by providing support in the form of concise, relevant, user-focused *critiques*.

1.2 The Need for Decision Support for Trauma Management

In his thesis, Rymon [75] presents a good argument for the usefulness of knowledge-based decision support in the trauma domain. His work and the system he implemented, TraumAID 2.0, served as the starting point for my own work in this area. TraumAID was designed to help physicians during the first few hours after an injury, a period often referred to as the *initial definitive management phase* of trauma care. This phase of care takes place in the hospital, after the patient has been resuscitated and stabilized and an initial assessment has been performed. It involves the performance of more definitive diagnostic tests to determine the extent of the patient's injuries, as well as some initial therapeutic activity.

As Rymon notes, there are several related ways in which computerized decision support can help with the management of patients:

- Supplementing physicians' expertise where it is lacking, such as in rural areas or outside normal working hours.
- Supporting standardization of care, which can help reduce health-care costs.
- Providing on-line Quality Assurance (Q.A.), by monitoring deviations from standards of care.

The TraumAID system, which has been under development at the University of Pennsylvania for over ten years, is a tool for assisting physicians during the initial definitive

management phase of patients with severe injuries [75, 92].[1] During this phase of patient care, 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, embodies a goal-directed approach to patient management. The system architecture 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 currently relevant goals. A more detailed discussion of TraumAID 2.0 appears in Chapter 2.

TraumAID 2.0's management plans have been retrospectively validated by a panel of three experienced trauma surgeons. The data used in this study were the records of 97 actual cases from the Medical College of Pennsylvania (MCP) Trauma Center involving injuries that were covered by TraumAID's knowledge base. The judges performed a blinded comparison of the actual care provided with the management plans produced by TraumAID 2.0 for the same cases. TraumAID's plans were preferred over the actual care to a statistically significant extent [18]. We thus believed such plans could provide a valid basis for producing critiques of physician plans which could lead to improvements in patient care.

1.3 The Problem: Getting Doctors to Use Computers

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 [3]. These observations have been made about systems that are designed to assist physicians *off-line*, not while they are actually engaged in a clinical task such as patient management. The problem is likely to be exacerbated when we attempt to integrate decision-support systems into the task environment to provide on-line support, which will be necessary if they are to be used in situations like trauma management, in which time is a critical factor.

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

For a decision-support system to be clinically viable it is essential not only that it produce valid advice, but also that its users can interact with it in a way that is both simple and that does not interfere with the task in which they are engaged. In the case of TraumAID this means solving both input and output problems.

To get information into TraumAID, we envision an electronic version of a standard *trauma flow sheet* (TFS). Normally in the trauma team, one member is a nurse who functions as a scribe, documenting all the findings, tests, and treatments in chronological order for the record. The multiple page trauma flow sheet has designated areas for specific information, such as demographics, mechanism of injury, physician response times, trauma score, Glasgow coma score, vital signs, location of wounds, results of primary assessment, intravenous therapy, diagnostic and therapeutic procedures, medications given, fluid intake and output, and disposition. Using our electronic TFS, relevant information entered by the scribe nurse will be automatically passed to TraumAID. Most importantly for TraumAID, all procedures ordered, clinical findings, test results, and therapeutic actions done will be available from the electronic TFS.

This dissertation is concerned with the problem of the system's output. 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. This plan 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, autonomous 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 she 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 she decides to go against TraumAID's proposals, the physician is, in effect, on her 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.

If we are interested in identifying problems with the intended management plan, the obvious first step is to identify what that plan is. Fortunately, in the trauma management situation, we can exploit the fact that many procedures require resources and equipment that must be *ordered* prior to carrying them out. In fact, those procedures that are most significant in terms of their cost and effect on the patient, and thus most important to identify if they are in error, are also those that are most likely to be ordered ahead of time. These orders will be recorded on the electronic TFS and so will be available to TraumAID. Since there is a gap between the time actions are ordered and the time they are carried out, it is possible for orders to be rescinded. By pointing out problems with orders (or missing orders), it should therefore be possible to have a clinically significant effect on patient management.

1.4 Statement of Thesis

In this dissertation, I will present an approach to real-time decision-support that uses a *critiquing* interface to produce advising behavior in a computer system. The approach I present is particularly appropriate for decision-support in domains that have the following features:

1. **Multiple goals**: Trauma management often involves reasoning about multiple interacting goals, some of which may be addressed together using a single procedure. Therefore, the physician's proposed actions must be evaluated globally, within the context of the entire plan. 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 she 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.

Similar approaches have been proposed for decision-support in domains such as aircraft and marine control, power plant operation, and complex process control tasks [37, 73].

In Chapter 2, I present an argument 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 claim that the ability to critique effectively depends on a combination of integrated knowledge structures and reasoning capabilities:

1. A plan recognition 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 her 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 critiques can be provided by a computer system through the description of TraumaTIQ, the critiquing module that I have implemented as part of the TraumAID system for trauma management [29, 30, 31, 32]. TraumaTIQ monitors trauma management in real time and produces comments when it detects potential errors that may have a negative effect on the outcome of the case.

In addition to the implemented system, the contributions of this work include:

- A situated approach to incremental plan recognition that is appropriate to the recognition of complex plans involving multiple goals and information seeking actions.
- An exploration of the relationship between planning and plan recognition systems in terms of the knowledge representation necessary to do both, and the process of adapting a planning system for plan recognition.
- A generic classification of erroneous actions from the perspective of their potential impact on the outcome of the plans in which they participate, and a methodology for evaluating the significance of errors for the purposes of critiquing.
- An evaluation of the results of applying TraumaTIQ retrospectively to the management of 97 actual trauma cases.

1.5 Dissertation 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 discusses previous approaches to plan recognition and describes the approach I have taken in TraumaTIQ. Chapter 5 introduces the notion of evaluating a plan and shows how my system determines those items that will be included in the final critique. In Chapter 6 I discuss some of the

issues involved in realizing a critique in natural language. Chapter 7 describes my implementation of TraumaTIQ, the critiquing module for TraumAID. Chapter 8 describes the results of applying the system to actual management plans. Finally, Chapter 9 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

The first version of TraumAID system (TraumAID 1.0), running on a laptop PC, was introduced experimentally into the Emergency Room at MCP for a 15 month period in 1988-1990. To use it, physicians had to leave the patient's bedside, which itself discouraged timely data entry. In addition, physicians objected to having to see TraumAID's entire management plan, noting that

- Much of the time, TraumAID's recommendation coincided with the physicians' own plans.
- It was difficult for physicians to determine what elements of the plan they should focus on to get information that would actually help them improve their overall performance.

In a sense, TraumAID was presenting both *too much* information – including items that the physicians were already well aware of – but also *too little* information – failing to indicate where the physicians should focus their attention to improve their plans.

These reactions led us to explore the possibility of an interface that could focus its output on just those items that are both *relevant* to the intentions of the physician, and *clinically significant* to the case at hand. This would address the two problems noted above by (a) taking account of the physicians' plans and (b) only presenting information that

they should therefore focus on. The approach we decided on is based on the critiquing model first introduced by Perry Miller [54] in his work on the ATTENDING system.

2.1 The Critiquing Paradigm

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 [81]). Their common feature 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.

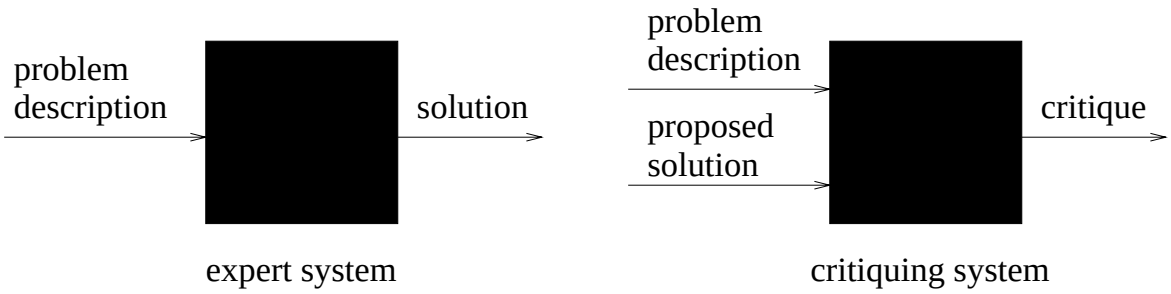

Figure 2.1: The expert system model vs. the critiquing model

In this dissertation, 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 [15].) However, unlike other work in this area, which has focused on the interaction between an "expert" system and a "novice" user, the model of cooperative communication my work is

based on is the interaction of 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.

As we will see in the next Chapter, the critiquing approach has been applied to a wide range of domains and problems. Some of these, like trauma management, are procedural in nature and involve developing a plan or specification for action, while others are more declarative and are concerned with producing a design specification. The critiquing methodology I will present in this dissertation is developed for the former type of problem. But while the problems are somewhat different, particularly in the types of constraints they are concerned with (e.g. temporal vs. spatial), there are also a number of similarities among problem domains that are suitable for critiquing.

In this Section, I consider three issues:

1. What features of the domain and the system are required for critiquing to be possible?
2. When is critiquing the preferred approach to communicating information to users?
3. What features of the domain influence *how* critiquing is done?

2.1.1 Requirements for Critiquing

To make critiquing possible, three basic features are necessary:

1. The domain must be modelled to the extent that the constraints on solutions that the critique will support have been defined. This may be done by enumerating *a priori* those constraints to be used in evaluating solutions, or by creating a planner (or other solution generator) whose output can be compared to the user's proposed solution at run time.
2. The actions peformed by the user must be sufficiently accessible to the system. Suchman's [84] example of an expert help system for a photocopier shows how information that is crucial to understanding the users' behavior may be conveyed by actions, such

as spoken comments, that the system does not have access to. There will always be actions that users perform that cannot be sensed by the systems they are using, but the more information the system has about what the user is doing, the less likely it will be to misinterpret the actions it knows about.

Related to this is the question of grain size. Systems that sense user actions at the level of individual operations on the machine have a different problem of interpretation than those that get compound action descriptions as input.

3. The user's proposed or intended actions must be accessible to the system, either by explicit notification or by prediction based on what has already been done. This is necessary for on-line critiquing, where the purpose is to *prevent* errors from being committed. If a *post hoc* evaluation is sufficient, then it is not required that the system know the user's intentions before they are carried out.

2.1.2 Motivations for critiquing

In general, there are several advantages of using a critiquing approach for decision-support rather than the more standard expert system approach [54]:

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 her 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 domain of trauma management seems to be a promising candidate for the critiquing approach. First, from the point of view of system acceptability it is important to keep in mind that physicians engaged in the management of trauma patients have multiple concurrent demands on their attention and must be able to make crucial decisions under time pressure. A system that presented its recommended plan and required the physician to interpret it in terms of her own reasoning about the case could be rejected on the grounds that it was too distracting or presented too much irrelevant information.

Another important factor is the level of experience of the intended users. The critiquing mode of interaction was designed as a way of communicating information to users who have some expertise in the domain of the system. Having some background and experience, these users may be prone to interpret an advice-giving system as an attempt to replace their judgment with that of a machine. The more prescriptive the advice produced by the system, the worse this effect will be. To address this problem, critiquing was proposed as a way of providing a focussed discussion of the physician's proposed actions, serving to remind her of items she may have overlooked while refraining from explicitly telling her what to do.

On the other hand, users with slightly less experience who may not be completely confident in their decision-making abilities may be tempted, given a system that provides them with a suggested or recommended solution, to follow that solution blindly without

questioning or understanding it. In a medical context this would be unacceptable, and a critiquing approach will also remedy this type of problem by requiring the user to develop her own plan initially without the help of the system.

The suitability of critiquing for communicating advice to experienced users means, however, that it may not be the best approach when dealing with problems in which the intended user is a novice in the domain and where the primary objective is to accomplish the task at hand rather than to teach or provide experience to the user. In such circumstances, the system serves as a guide or director and would do better to give direct suggestions for action rather than critiquing.

The inherent *flexibility* of a critique arises from the fact that it does not have to commit to any particular solution to the problem it is presented with. Rather, it may accept a range of solutions as long as they satisfy the relevant constraints reasonably well. This feature is particularly applicable to the problem of upholding *practice guidelines*. Currently in the medical community a great deal of attention is being paid to the specification of guidelines or protocols for patient management.[1] These guidelines are underspecified, *skeletal plans* [77], which must be fleshed out by the physician during the course of practice. In this sense, TraumAID's knowledge base can be taken as a validated set of guidelines for trauma management.

This view of plans is similar to that put forth by Lucy Suchman [84], who argues that rather than interpreting observed actions as being determined *a priori* by a plan, it is essential to consider the specific context in which actions are executed in order to understand them. In this view, plans are necessarily vague because they cannot possibly specify every detail of situated action at the executable level. Plans are taken more as guiding action on an abstract level rather than determining it concretely.

This description fits the case of medical practice guidelines perfectly – there is much that is left out of the guidelines and left to individual physicians to fill in simply because it would be impossible to specify every possible contingency to the level of detail necessary for action. Furthermore, it is not the guideline designers' purpose to specify actions that precisely. Rather, the purpose of a guideline is to insure that certain goals are addressed

[1] Guidelines are created for the purposes of quality assurance and provide general specifications for how to proceed in given situations. Protocols include more specific rules to follow and are used to insure comparable samples in controlled clinical trials.

and that certain constraints are satisfied. This is a task for which critiquing is well suited because, like the guideline, it does not seek to direct action but simply to constrain it. Following Suchman's description of situated action, the critiquing system I have developed interprets actions *in context* and generates its responses in consideration of the current situation.

2.1.3 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 [54]. 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.

2.1.4 Miller's Two Dimensions

The first dimension is the degree of *independence of decisions*. Miller shows that treatment plans in the domain of 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 [45]. 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.[2]

The critiquing model I am developing is designed for 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.[3] 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. This 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]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.

[3]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, breathing, circulation, etc.

2.1.5 Two Additional 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 [22]. 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 [22], almost all decisions are based on preferences, and the critiquing process basically involves pointing out when certain constraints have not been satisfied. 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 she 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.

2.2 An Analysis of Judges' Error Data

In order to discover what constitutes an error in trauma management, I first looked at the data from the retrospective evaluation of TraumAID's performance as a management planner [18, 75] (see Chapter 1). 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 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.

In order to use these evaluations to classify errors, each judgment was transformed into a "snapshot" of the case up to that point. These snapshots consisted of a list of attributes, 242 in all, including information about the wounds, the findings determined, and actions performed so far. The findings attributes had three possible values: yes, no, and unknown. The action attributes were binary: done or not done.

The snapshots were classified into ten categories for each possible action, α (following each item in parentheses is the marking used to indicate that class):

1. Doing α in this situation is an error of *commission* (C).

2. Doing α in this situation is a *critical* error of commission (C*).[4]

[4]Errors were classified as critical if they lead the judge to mark the case in the lower two categories of acceptability.

3. *Not* doing α in this situation is an error of *omission* (O).

4. Not doing α in this situation is a critical error of omission (O*).

5. α was done too early (E).

6. α was done critically too early (E*).

7. α was done too late (L).

8. α was done critically too late (L*).

9. Doing α is not an error (NIL).

10. *Not* doing α is not an error (ONIL).

The final category was added for completeness, and it was assumed that whenever a judge did not specifically mention an action that was not part of the plan, it was not considered an error not to do that action. The snapshot data were then split up into separate files for each action in the domain.

Figure 2.2 shows the frequency of error judgments for the 35 out of 153 available actions about which a total of 10 or more error judgments were made by all three judges. These 35 actions became the focus of the remaining analysis.

A subset of the data files for the actions which tended to draw more error judgments was chosen for analysis. The classification program IND was used to generate classification trees for these actions. By running IND on separate data for individual actions, the trees produced represent predictions of error judgments for those actions separately. For example, the tree constructed for peritoneal lavage (Figure 2.3) indicates the conditions under which a lavage would be considered to be an error of commission (if it was done) or an error of omission (if it was not done). Each node in the tree represents possible values of a single variable (N = Negative, Y = Positive, U = Unknown). The leftmost (highest) nodes are the most discriminating regarding the action in question. The leaves indicate the frequency of each error type in order [O, ONIL, C, C*, O, O*, E, E*, L, L*], followed by the most likely judgment based on those frequencies.

According to the peritoneal lavage tree, the best indication of whether it is right to do a lavage is whether or not the patient has abdominal scarring. This makes sense because

	C	C*	O	O*	E	E*	L	L*
Abdomen Exam	0	0	40	81	0	0	0	0
Abdominal X-ray	5	2	1	4	0	0	0	0
Abdominal Scarring	20	0	1	0	0	0	0	0
Absent Rectal Tone	6	3	9	12	0	0	0	0
Antibiotics	44	37	14	5	1	0	7	2
Any LWE	6	8	10	5	0	0	0	1
Blood In Gastrointestinal Tract	0	0	18	29	0	0	0	0
Blood In Stool	1	3	12	13	0	0	0	0
Chest Exam	0	0	10	38	0	0	0	0
Cover Wound L Chest	0	0	3	4	2	2	5	3
Cover Wound R Chest	0	0	1	1	2	1	7	2
CT Scan Abdomen	3	5	3	1	0	0	0	0
Decr. Breath Sounds (Left)	0	0	13	7	0	0	0	0
Decr. Breath Sounds (Right)	0	0	13	7	0	0	0	0
Distended Neck Veins	0	0	10	14	1	1	31	4
IVP	5	11	0	17	0	0	0	0
Laparotomy	0	7	0	5	0	0	0	0
Muffled Heart Sounds	0	0	25	26	0	0	0	0
Neurologic Exam	4	0	39	43	0	0	0	0
Ng Aspiration P Lat Cxr	2	1	11	10	0	0	0	0
Objective Abdominal Evaluation	0	0	4	7	0	0	0	0
Pericardial Tamponade	0	0	12	70	0	0	0	0
Peritoneal Lavage	12	27	1	3	1	0	0	0
Physical Exam	0	0	2	10	0	0	0	0
Questions Re Heart Injury	0	0	32	26	0	0	0	0
Re-Evaluate Abdomen	0	0	12	3	0	0	0	0
Survey Chest Xray	5	2	5	40	2	1	9	8
Tenderness	4	5	3	4	0	0	0	1
Thoracic Aortogram	6	4	0	1	0	0	0	0
Thoracotomy (Left)	0	5	2	1	0	0	0	0
Thoracotomy (Right)	0	3	2	0	0	0	0	0
Urinalysis Rbc	4	2	11	5	0	0	0	1
Urologic Consult	3	0	0	0	0	0	0	0
Vascular Injury	0	0	0	6	0	0	0	0
Verify Hidden P Asp Sac	0	0	0	0	0	0	0	0

Figure 2.2: Frequency of judge's error comments on selected actions.

```
Percentage accuracy for tree 1 = 95.271 +/- 0.720864
Mean square error for tree 1 = 0.0768723
Expected accuracy for tree 1 = 95.271
Average utility for tree 1 = 96.5975
Estimated accuracy from X-val = 93.951+/-0.80962
Leaf count for tree 1 = 15, expected = 15.000000

ABDOMINAL-SCARRING in Y,U:
|   NG-ASPIRATION in N,Y:
|   |   LAPAROTOMY = U:
|   |   |   WOUND1-YLOC in LC,UA,LA:
|   |   |   |   WOUND1-ZLOC in POST,LAT: [+0+0+1+2+0+0+0+0+0+0]  C*
|   |   |   |   WOUND1-ZLOC in ANT: [+15+0+0+2+0+0+0+0+1+0]  NIL
|   |   |   WOUND1-YLOC in UC: [+0+9+0+0+0+0+0+0+0+0]  ONIL
|   |   LAPAROTOMY = Y: [+0+18+0+0+0+0+0+0+0+0]  ONIL
|   NG-ASPIRATION in U:
|   |   BLOOD-IN-STOOL in U,N: [+3+503+0+0+0+1+0+0+0+0]  ONIL
|   |   BLOOD-IN-STOOL in Y: [+3+0+0+0+0+0+0+0+0+0]  NIL
ABDOMINAL-SCARRING in N:
|   LAPAROTOMY = U:
|   |   TENDERNESS in U,Y:
|   |   |   WOUND1-ZLOC in ANT,LAT: [+94+0+4+13+0+0+0+0+0+0]  NIL
|   |   |   WOUND1-ZLOC in POST:
|   |   |   |   OBTUNDATION in N,U: [+3+0+4+5+0+0+0+0+0+0]  C*
|   |   |   |   OBTUNDATION in Y: [+6+0+0+3+0+0+0+0+0+0]  NIL
|   |   TENDERNESS in N:
|   |   |   OBSERVE-BLEED-PERIT-ABD = U:
|   |   |   |   REASSESS-L-CHEST = U:
|   |   |   |   |   REASSESS-R-CHEST = U:
|   |   |   |   |   |   NG-ASPIRATION in U,Y: [+19+0+1+1+0+0+0+0+0+0]  NIL
|   |   |   |   |   |   NG-ASPIRATION in N: [+0+0+2+1+0+0+0+0+0+0]  C
|   |   |   |   |   REASSESS-R-CHEST = Y: [+0+3+0+0+0+0+0+0+0+0]  ONIL
|   |   |   |   REASSESS-L-CHEST = Y: [+0+15+0+0+0+0+0+0+0+0]  ONIL
|   |   |   OBSERVE-BLEED-PERIT-ABD = Y: [+0+27+0+0+1+2+0+0+0+0]  ONIL
|   LAPAROTOMY = Y: [+0+105+0+0+0+0+0+0+0+0]  ONIL
```

Figure 2.3: The tree produced for peritoneal lavage.

# judges making comment			
	1	2	3
Commission	112	21	5
Omission	288	95	15

Table 2.1: Agreement between judges on actions

a lavage should not be done if the patient has a scarred abdomen. In addition, since lavage is an abdominal procedure, all of the decision points on this tree have to do with symptoms in the abdomen, or with the location of the wound being in the abdomen or lower chest. A final point about this tree is that the decision to do a laparotomy (operate on the abdomen) obviates the need for a peritoneal lavage, since any information the lavage would have provided will be made evident during the operation. Therefore, the decision *not* to do a lavage when a laparotomy is called for is always put in the category ONIL (not a mistake not to do it).

I generated trees for five actions of different types, each of which had a high number of errors associated with it in the judges evaluations. These actions were: peritoneal lavage, survey chest X-ray, antibiotics, neurologic exam, and checking for distended neck veins. The tree for peritoneal lavage has the highest accuracy of all the trees with an estimated accuracy of 93.951%, and is the most intuitively understandable tree. The lowest estimated accuracy for the actions I analyzed was 85.723% for the antibiotics tree.

The low predicted accuracy of these trees indicates a lack of agreement between the judges as to what constitutes an error. In fact, as table 2.1 shows, the majority of comments made regarding errors of commission and omission were made by only one judge. For errors of commission, only 21 comments were produced by two of the three judges and only 5 comments were produced by all three judges, as compared to 112 comments that were made by one judge alone. In the case of omissions, 288 omissions were noted by a single judge, 95 by two judges, and just 15 by all three judges. There were no agreements on errors of scheduling.

Even more problematic than the low accuracy is the fact that an error of any kind is only predicted by a total of five branches in three of the five trees. Since the number of errors marked is so small compared to the overall number of actions, and there is little

agreement about the errors that *are* marked, it is generally more accurate to predict that an action (or lack of action) is not an error.

It is probable that the lack of predictive power in these trees is due in large part to the fact that the error data were not collected with this analysis in mind, but rather for the purposes of an overall evaluation of the TraumAID system. The judges were not asked to mark down every item they believed was an error, but were supposed to use their individual markings to guide them in picking an overall ranking for the case. Looking at the evaluation forms, it appears that the judges sometimes did not mark down any errors, even when they gave the case a low rating.

It seems, therefore, that additional data would be necessary in order to find out anything useful about the classification of errors in trauma care by this method. This analysis led me to consider that rather than trying to define a classification scheme for errors from scratch, it would be more productive to use the plans generated by TraumAID as a gold-standard, comparing them to the physicians' intended plans and considering any discrepancies as potential errors. Treating TraumAID's plans as a gold-standard for care is justified because they have been judged to be globally acceptable by a panel of experts.

2.3 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 2.4). Using this approach, TraumAID is able to flexibly interleave diagnostic and therapeutic action, thus addressing the competing needs arising from multiple injuries.

The representation of plans in TraumAID and TraumaTIQ will be discussed in more detail in Chapter 4. I will just outline it briefly here.

TraumAID's plans are constructed out of three types of objects: *goals*, *procedures*, and *actions*. Goals correspond to abstract plans which are addressed by performing actions in the world. TraumAID's rule-based reasoner determines what goals are relevant to pursue, given the current state of the patient, and its planner then constructs a plan consisting of actions to address the relevant goals.

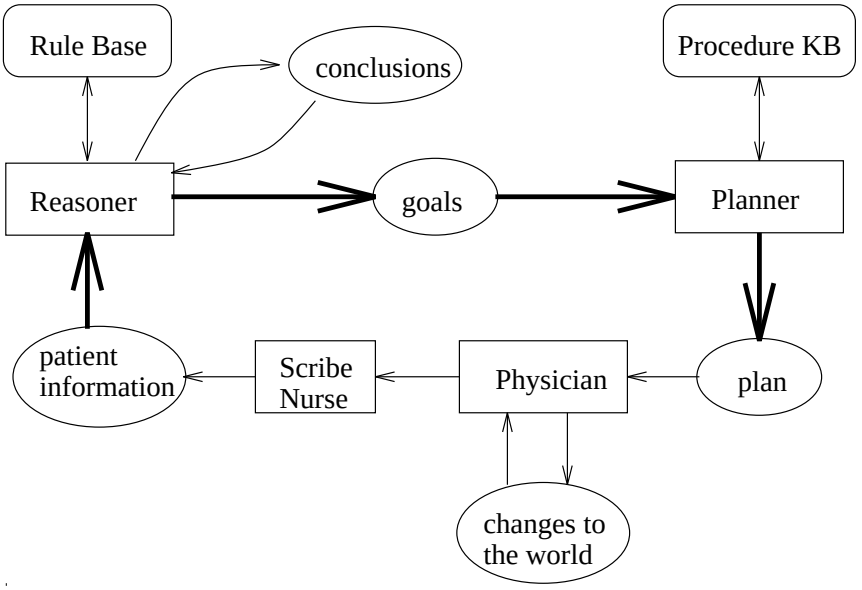

Figure 2.4: System Architecture of TraumAID 2.0

TraumAID associates each goal with a disjunctive list of procedures for addressing it. This association is called a *goal-procedure mapping*. In each mapping, the procedures are ordered preferentially with respect to the goal. When planning to address a *set* of goals, TraumAID's planner selects one procedure for each goal from the goal-procedure mapping. Selection depends on both the local procedure preference rankings for each goal, and global considerations resulting from the need to address multiple goals concurrently. In the situation shown in Figure 2.5(b), procedure P1 can be used to address both relevant goals, G1 and G2.

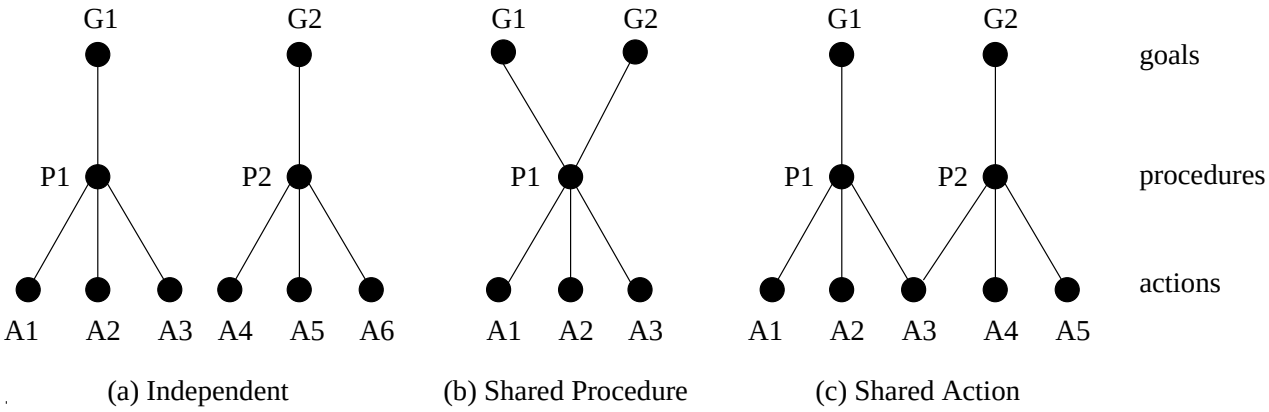

Figure 2.5: Three possible multiple goal-procedure-action configurations

A procedure comprises an ordered sequence of actions, stored in a *procedure-action mapping*. Actions can participate in more than one procedure at a time, and thus can be used in addressing more than one goal as shown in Figure 2.5(c). This concept of *action overloading* is exploited by TraumAID's planner to produce globally efficient plans.

To determine the currently relevant goals for the planner to address, TraumAID's reasoner 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.

The reasoner controls information acquisition using a conservative, *staged* strategy for selecting diagnosis and treatment goals [75]: goals whose satisfaction requires expensive and/or risky, 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. These strategies appear in the knowledge base as a chain 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. Goals may also be related in a plan by being addressable by the same action. For example, the goal of RO-HEMOTHORAX ("rule out blood in the chest cavity") leads to doing a survey chest x-ray, as does the goal of RO-LACERATED-DIAPHRAGM ("rule out a lacerated diaphragm"). There is no strategic relationship between these two goals – the injuries may be caused by different wounds – yet they are related by the fact that the same action can be used to address them.

Once a set of relevant goals has been determined, the planner's choice of how to address each goal is based on both local and global considerations. Given a goal's *goal-procedure mapping*, all else being equal, the first procedure will be chosen. A less preferred procedure may be selected if it would result in a globally more optimal (less costly) plan. Figure 2.5 shows three possible configurations for a portion of a plan addressing two goals.

The plans produced by TraumAID's planner are partially ordered according to both logistical and clinical constraints. Logistical constraints are due to the fact that patients are only moved in one direction through the Trauma Center – from the emergency center, optionally to the radiology suite, then optionally to the operating suite, and finally either to the trauma unit or discharged to go home. Actions with logistical constraints on their performance are scheduled so as to avoid transferring the patient back to a place he has already been.

Clinical constraints have to do with the *urgency* and *priority* of an action, which it inherits from the goals it is being used to address. The urgency can be either *catastrophic*, *unstable*, or *stable*, representing the patient's condition and thus how quickly the goal must be addressed. Catastrophic goals must be addressed immediately. Unstable goals must be addressed before stable goals. Priorities represent standard practices of trauma care: if there are no differences in urgency, problems involving the airway are addressed before those involving breathing, which are addressed before those involving circulation, etc. (the "ABC's of trauma care").

The planner initially uses a "greedy" algorithm that iterates throughout the current set of goals ordered by urgency and priority, selecting a procedure to address each one not addressed by an already selected procedure and then ordering those procedures 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 [74, 75]. Only the beginning of the plan is optimized because complete optimization is NP-hard and because later parts of the plan may change when new information is recorded.

2.4 The Trauma Team

The hospital personnel responsible for the initial definitive management of trauma patients are referred to as the "trauma team." A trauma team consists of five to seven people, including both physicians and nurses. For the purposes of TraumAID we are primarily interested in two members of the trauma team:

- The *chief surgical resident* is the primary decision maker on the team[5] and as such is the person whose decisions TraumAID and TraumaTIQ are intended to influence. In the rest of this dissertation, the chief surgical resident will be referred to simply as "the physician."

- The *scribe nurse* is responsible for documenting all the findings, tests, and treatments in chronological order for the record. Normally, these data are recorded on a paper form called a Trauma Flow Sheet (TFS). When TraumAID is in use, the scribe nurse will enter information directly into the computer instead.

2.5 An Architecture for Critiquing Trauma Management Plans

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.

[5] Although she is supervised by a faculty attending physician.

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 she 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.

TraumaTIQ's critiquing process is triggered whenever new information is entered by the scribe nurse and delivered to TraumAID. This information can be in the form of (1) bedside findings, (2) diagnostic test results (indicating both that a diagnostic action has been performed and what the results of that action were), (3) therapeutic actions performed, or (4) diagnostic or therapeutic actions ordered by the physician. From TraumaTIQ's point of view, the orders represent the actions that the physician *intends* to perform, and therefore provide the basis for formulating a critique.

Given some new input to the system, TraumaTIQ then interprets the physician's orders in terms of their underlying goals, evaluates the inferred plan structure by comparing it to TraumAID's recommended plan, and finally generates an English critique addressing elements of the physician's plan that were found during the evaluation phase to represent potential problems. Figure 2.6 shows the architecture of TraumaTIQ, comprising *plan recognition*, *plan evaluation*, and *critique generation*.

The purpose of plan recognition is to understand *why* the physician is doing what she is doing. This enables both *suggesting alternatives* to address the same goals, and *explaining* why actions may not be justified. The plan recognition module will be described in detail in Chapter 4.

Plan evaluation is done to identify errors and determine whether they are significant enough to report in the critique. Potential errors are identified by looking for discrepancies between the physician's plan and TraumAID's plan. Errors may be due to omissions, commissions, or violations of scheduling constraints in the physician's plan. The output of the plan evaluator is the *informational content* of the critique. The plan evaluation module is described in detail in Chapter 5.

Finally, the critique generation module takes the output of the plan evaluator, organizes the comments by topic, and generates English sentences by filling slots in pre-written templates with the relevant concepts. This process is described in Chapter 6.

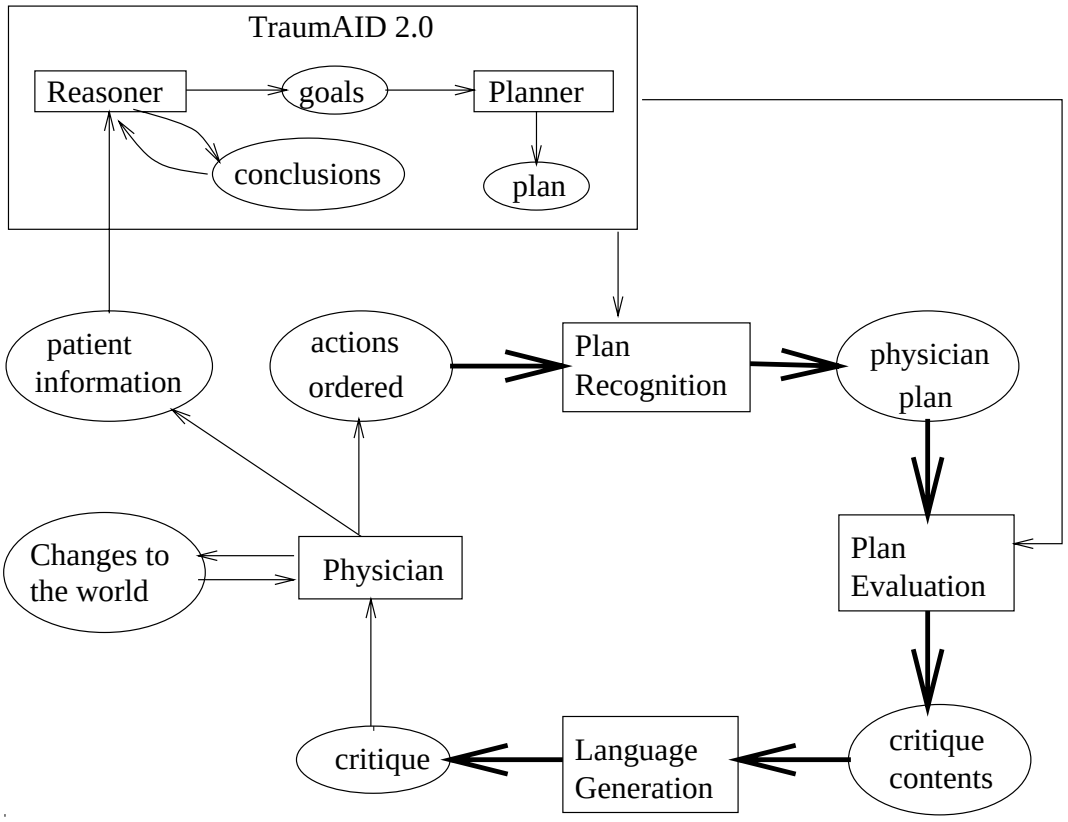

Figure 2.6: The TraumaTIQ module

Before going on to describe the components of the critiquing architecture in depth, the next chapter provides some background on work in critiquing and related areas.

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. Critiquing systems have been developed for diverse problems in a number of domains, ranging from medical therapy planning to computer aided design. Other systems perform some of the functions of critiquing systems, although their developers do not call what they do "critiquing." In this chapter, I present a review of some of the literature relevant to critiquing and discuss their contributions and weaknesses.

3.1 Medical Critiquing Systems

The term critiquing was first introduced by Perry Miller [53] in describing his ATTENDING system for critiquing anesthesia management. Since then, much of the work on critiquing has been done in medical domains. Providing decision-support to physicians lends itself to the critiquing approach both because of the need for a user-focused, non-confrontational style of interaction, and because of the variability inherent in solutions to medical problems. In this section, I describe several medical critiquing systems, starting with the original ATTENDING systems.

3.1.1 The ATTENDING family of critics

The original ATTENDING system [53], 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 she 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 sub-decisions. 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 [54].

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 she 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 she *does the right thing*.

3.1.2 ONCOCIN: User-guided critiquing

The ONCOCIN system [45] 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 she 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 she 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 she is specifically interested, it has the disadvantage that she 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.

3.1.3 HyperCritic: Critiquing from Automated Medical Records

The HyperCritic system developed by Van der Lei [90] 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.

3.1.4 Critiquing Guideline-Based Care

In [77] Shahar and Musen discuss the representation and reasoning needed to support the application of clinical guidelines and protocols to patient care. These standardized policies for care are used both to ensure the quality of care and (in the case of protocols) to facilitate experimental analysis. These guidelines can be thought of as highly reusable skeletal reactive plans which must be refined over time in order to be used. They represent these plans using language of temporal-abstraction patterns which describe events and parameters that hold during intervals of time. The application of clinical guidelines can be problematic because the guidelines are often ambiguous or incomplete, and require the physician to interpret the intentions behind the guideline in order to decide how to act.

Shahar and Musen characterize the relationship between the physician and the automated decision-support system as a collaboration between two planners, each of which may bring different knowledge and skills to the situation. They emphasize the importance of plan recognition and reasoning about plan revision on the part of the system in order to

understand the relationship between the intentions of the designer of the clinical guideline and the physician executing the guideline. When the executed treatment deviates from the guideline, it is important for the system to be able to determine whether the deviation upholds the *spirit* of the guideline, achieving the designer's intended goals, or whether it represents a real violation of the guideline. In order to do this, it is necessary that the actions and plans in the guideline be *annotated* with the intentions behind them, a feature that is not currently available in clinical guidelines. The system must also have a representation of actions and plans in the domain so as to be able to infer the goals underlying the executing physician's actions.

The work presented in this dissertation addresses a number of the issues presented in [77]. The information in TraumAID's knowledge base represents guidelines for trauma management, which are refined incrementally over time by TraumAID's planner. The role of TraumaTIQ is to interpret the actions of the managing physician and determine when they deviate significantly from TraumAID's guidelines. Chapters 4-7 describe the approach to this problem that I have developed and implemented in TraumaTIQ.

3.2 Critiquing Designs

Another area in which the critiquing paradigm has been applied with some success is that of design critiquing [23, 43, 72]. 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 [43] 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 [22]. 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 [22] 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 [24] 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 are concerned with is not covered.

3.3 Critiquing Based on Cognitive Biases

In his research on developing automated critics, Silverman [79] has developed what he calls a generative theory of "bugs" to produce a rule base of errors resulting from cognitive biases and slips or lapses in memory. This work draws heavily from work in cognitive psychology on identifying the causes of human error [41]. According to this theory, people often exhibit judgment biases in the performance of cognitive tasks, causing them to commit errors. These errors occur in tasks involving information acquisition, information processing, intended output, and paying attention to feedback.

[1] This is similar to the Program Enhancement Advisor described in [60] except that the latter is more concerned with the knowledge representations necessary to explain the system's reasoning.

This rule base of errors 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 [79], forecasting potential threats to 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 she 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.

3.4 Language generation and critiquing

One of the key issues for any system that is intended to interact with users is how to communicate its information as effectively as possible. In the case of critiquing, this has generally been done through natural language. The generation of critiques as coherent natural language texts has been addressed most thoroughly by Rankin [70]. A discussion of both deep and surface requirements of natural language critiques appears in [55], which compares the linguistic tasks of explanation and critiquing. The following subsections describe these papers.

3.4.1 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 [70] 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 she 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 generated. 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,

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

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) [51] 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.

3.4.2 The relationship between explanation and critiquing

In their analysis of the attitudes of physicians toward computer-based consultation systems, Teach and Shortliffe [86] 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 [60]. More recently, more sophisticated approaches have been proposed, integrating work from the natural language generation community with work on explainable expert systems [61], and allowing for interactive explanation, where the user can request additional clarification or justification in response to the system's explanations [57].

In [55], 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 [5].

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 6.

3.5 Reminders and Alerts

Another approach to on-line decision support that is related to critiquing is represented by reminder or alerting systems [7, 52]. 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 accommodating 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 unnecessary comments, which may be the reason that physicians using the HELP system [7] 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 physician, thus adding a level of indirection between the system and the primary care-giver.

3.6 Related work in Human-Computer Interaction

The field of Human-Computer Interaction (HCI) has in recent years started to address the problem of assisting operators of increasingly complex technological systems. As systems become more complex, it is no longer possible to rely on system design and automation to reduce the occurrence of human error. Rather it is necessary to develop systems that can interpret their operators' performance, identify errors, and respond appropriately either by providing necessary information or by automating certain parts of the task in progress.

To this end, architectures for an intelligent operator support interfaces have been proposed by Hollnagel [37] and by Rouse et. al. [73]. The former is primarily an implementation of a taxonomy for error classification that I will present in detail in Chapter 5. I will discuss the latter here because it is a more comprehensive high level approach to the problem.

Rouse et. al.'s architecture is intended primarily for the task of fighter aircraft operation, although its applicability in other domains such as marine operations, power production, and process control is mentioned. The interface architecture comprises four main modules, all of which have access to representations of world, system, and operator state:

- The *operator model* develops and maintains a model of the operators plans and goals based on her observed actions. In addition, it predicts the resources she will need based on her intended actions, and provides an overall evaluation of her performance.
- The error monitor identifies and classifies errors, and responds to errors either by monitoring their existence, supplying feedback, or recommending that the system take over.
- The adaptive aid module decides how to respond to errors, either by trying to make the task easier for the human operator, by dividing the task between the system and the operator, or by automating the task entirely.
- The interface manager handles the communication between the system and operator, scheduling comments and requests for information.

While this architecture and TraumaTIQ have several high-level features in common, on a lower level they are quite different. The primary difference is the fact that Rouse et.

al.'s architecture is designed for an interface that has complete access to the system state and operator actions, and can actually affect the system directly. TraumaTIQ, on the other hand, only has indirect access to information about the state of the patient and the physician's actions, and it definitely *cannot* affect the patient directly. While a majority of the effort in the work on operator support has gone into determining when the system should *take over* from the user, TraumaTIQ is much more dependent on the ability to infer what the physician is doing on the basis of incomplete or uncertain information, and how to respond in such a way as to correct or avoid errors. Thus, the plan recognition technique used by TraumaTIQ is more sophisticated than that used in the operator model described above.

In addition, since it is impossible for TraumAID to take over the management of the patient, it is very important that the feedback generated by TraumaTIQ in the form of critiques is understandable by and acceptable to physicians. The primary concerns for TraumaTIQ's output interface are natural language issues: first, that explanations and justifications are provided, and second that the critique is phrased in such a way as to get a response from physicians, or at least not annoy them. Rouse et.al., on the other hand, focus on interface management as a resource allocation problem, involving various input and output modalities that must be available for the system's messages and requests to be processed.

3.7 Critiquing vs. Tutoring

A final body of work that is closely related to critiquing are intelligent tutoring systems. These may be static problem-solving tutors designed to teach reasoning skills such as diagnosis [4, 46, 50, 56], or training environments for teaching behavior skills in dynamic situations [19, 40, 48, 91]. The latter give students the opportunity to perform the task they are training on in a simulated environment that provides them with feedback as they learn and gain experience. Both critiquing systems and training systems involve monitoring the user, evaluating her performance, and responding on the basis of that evaluation. They both require some representation of expert domain knowledge, a facility for modeling the user's behavior, metrics for evaluating performance, and the ability to communicate with

users. The similarities have led recently to a number of efforts to combine critiquing and tutoring functions [25, 39], an idea that is also being pursued in the TraumAID project (see Chapter 9).

Any attempt to combine critiquing and tutoring should, however, be sensitive to the fundamental differences between the two functions. A critiquing system is designed to provide support for users who are already trained in the domain of application, and thus are primarily concerned with correcting errors in judgment or lapses in memory. Tutoring systems, on the other hand, are designed for novice users who may not be aware of the correct use of the techniques and skills they are learning. Therefore, tutoring systems must put more effort into representing students' potential errors and providing adequate explanation. For this reason, their domain is often restricted so that creating the knowledge base is manageable.

Critiquing systems, on the other hand, can use the assumption that users will often act correctly to minimize the amount of specialized *a priori* knowledge about errors that must be encoded. In addition, while explanation is an important part of critiquing, the level of detail required is much less than for tutoring since the users can be relied on to know the basics of the domain. It is also less important for a critiquing system than it is for a tutoring system to be able to confirm and reinforce the user's correct decisions. The added depth and breadth of output required for intelligent tutoring means that the problem of language generation is more complex in these systems.

3.8 Comparison of TraumaTIQ to other systems

As we have seen in this chapter, the critiquing approach has been applied to a wide range of domains and applications, from architectural design to Lisp programming to medicine. Table 3.1 is a summary of several medical critiquing systems, including Trauma-TIQ, according to a number of features that are relevant to the problem of providing effective decision support for quality assurance.

Most medical critiquing systems have been designed to critique *therapeutic* management plans, and thus they assume that the patient's diagnosis is known, and that therefore the relevant therapeutic goals are shared by the physician and the system. The exception to

System	Critiques Therapy	Critiques Diagnosis	Critiques Scheduling	Real-Time Performance	Output Generation
ATTENDING	Y	ICON	N	N	ATN
ONCOCIN	Y	N	N	N	Rule Trans.
CLAS	N	N	N	Y	Templates
Reminder	N	N	N	Y	Canned Text
HyperCritic	Y	N	N	N	Templates
Rankin	Y	N	Y	N	RST
TraumaTIQ	Y	Y	Y	Y	Templates

Table 3.1: Summary of Critiquing Systems

this was one of Miller's ATTENDING critics, called ICON [54], which was designed to critique differential *diagnoses*, but not in combination with therapy planning.

In contrast, TraumaTIQ starts with the diagnosis being unknown and critiques both diagnostic and therapeutic actions. TraumaTIQ does not critique diagnostic conclusions directly, but to the extent that incorrect diagnostic reasoning leads to incorrect action, that reasoning is critiqued as well.

Very few of the systems presented here reason about the temporal aspects of plans, and so they do not critique the *scheduling* of actions. In an urgent situation such as trauma, the scheduling of actions can be critical, and so TraumaTIQ addresses this issue as well by producing critiques when scheduling constraints are violated in the physician's plans.

A related issue is the question of *real-time performance*. Of the systems reviewed here, only the reminder and alert systems, which bring problems to physicians' attention *after* they manifest themselves, operate in real time. But these systems do not anticipate problems. Real-time performance is necessary for TraumaTIQ to be able to affect the outcome of a case that may unfold very quickly.

Finally, these systems use a variety of more or less sophisticated methods for generating natural language output. Some are focused on the surface generation of sentences by techniques such as ATNs or rule translations, while others, such as Rankin's RST-based generation, are more concerned with the global organization of text. TraumaTIQ currently uses a template filling procedure to produce English sentences, and does only a rudimentary organization of the text. Future work will address the issue of language generation in more detail.

Chapter 4

Recognizing the Physician's Plan

Intelligent interaction with another agent often depends on the ability to understand the agent's underlying *mental states* leading her to act as she does. These mental states include *beliefs* about the world, *desires* for the future state of the world, and *intentions* to act in certain ways. The process of inferring these mental states is generally referred to as plan recognition.

The necessity of making these kinds of inferences was first observed in work on story understanding [93] and cooperative response generation in dialogue systems [2, 9, 78]. In studies of naturally occurring dialogues it was observed that the respondent to a question would often provide information that was not explicitly requested, or correct perceived mistakes in the questioner's plan. This type of behavior, it was argued, could only be achieved with the ability to infer the plans that motivate questions that are asked.

Several systems have been implemented that make use of plan recognition to provide cooperative responses in dialogue. Plan recognition has also been used effectively in systems for text understanding [47, 35], intelligent computer-assisted instruction (ICAI) [49, 48], and automated help systems [6, 94]. In [34] Goodman and Litman present a comprehensive review of the uses of plan recognition for intelligent interface design, and discuss the scope and limitations of current plan recognition work.

Different approaches to plan recognition have tended to focus on recognizing different aspects of agents' plans. Many researchers have been interested in identifying an agent's beliefs given her intended actions (and possibly her desired goals as well) [2, 20, 65], in

order to identify and correct erroneous beliefs that might lead to faulty plans. Others have been concerned with predicting the agent's intended future actions in order to automate their execution [6]. Still others are interested in inferring the agent's desires so as to suggest alternative ways of achieving them [49], or to resolve ambiguities in dialogue interactions [44, 69].

The importance of plan recognition for automated decision support has been discussed by Shahar and Musen [77]. They point out that by inferring the goals underlying physician actions, a system can be more accommodating by accepting alternative ways of addressing goals. As long as the physician is pursuing a goal that is acceptable to the system, it may not be necessary to critique her behavior. In developing TraumaTIQ, I identified two additional factors motivating plan recognition in support of critiquing:

Explanation One of the key features leading to acceptance of a decision-support system is the system's ability to explain its reasoning [86]. In the case of TraumaTIQ, a critique of a proposed action should include an explanation of why the system thinks the physician is doing that action.

Proposing alternatives Understanding the goal(s) underlying a physician's action can allow the system to see the action as an alternative to what it would recommend for addressing that goal. If its recommendation has advantages in cost, speed, non-invasiveness, etc., it can be presented to the physician in this way, while supporting the physicians' original intentions.

Plan recognition problems vary according to a number of features [42]. First, can the inferring agent assume that the planning agent wants her 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, and the possible plans that may be pursued? Third, is it possible that the performing agent's plan may be erroneous or malformed in some way? The following section discusses three approaches to plan recognition that have appeared in the literature, reflecting different combinations of the above features.

4.1 Approaches to Plan Recognition

4.1.1 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 [21]. In the STRIPS formalism, a plan is represented as 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 may actually be pursuing that higher level plan. In language understanding systems [2, 9, 78], 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 in the discourse, 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 [47, 44, 64]. 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. In particular, Pollack [64, 65] 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.

4.1.2 A Formal Model of Keyhole Recognition

Kautz's plan recognition algorithm [42] is a model of keyhole recognition.[1] It relies on the assumptions that the system has complete knowledge of the plans in the domain, and that the agent's plans will always be correct in terms of achieving her goals.

Knowledge of plans is represented in a plan library comprising two interrelated hierarchies. The *abstraction hierarchy* represents inheritance relationships between plans and is composed of IS-A links between plan schemas. The *decomposition hierarchy* determines how the plans are executed. Each plan schema is decomposed into executable actions. In addition, the decomposition hierarchy contains information about action preconditions, effects, and constraints.

At the top of the plan hierarchy is a special plan type called an *End-Event*, which is not subsumed by or contained in any other plans. The goal of the plan recognition algorithm is, given the plan library and user input in the form of action instantiations, to construct an *explanatory plan graph* for those actions containing exactly one End-Event. The algorithm recognizes plans incrementally, incorporating new actions into the explanatory plan graph as they are input.

When there is more than one plan graph that will explain the observed behavior, the algorithm tries to resolve the ambiguity by minimizing the number of actions that are explained by different plan schemas. 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

[1] The metaphor here is that the observer is watching through a keyhole and attempting to recognize the plans of an agent who is not acting cooperatively so as to have her plans be understood.

assume that the *simplest* plan (i.e. the plan with the smallest number of 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.

4.1.3 Probabilistic approaches to plan recognition

Recently, interest has been increasing in applying probabilistic models to the problem of plan recognition. This interest is due in part to the observation that plan recognition is basically a problem of reasoning under uncertainty and so should be an appropriate application for techniques based on probability theory. A further motivation for this work is the increasing interest in keyhole, as opposed to intended, recognition. Intended recognition licenses a number of assumptions about the relationship between what is said and what is intended that are not valid in a keyhole recognition situation. Lacking these simplifying assumptions, the observer's ability to disambiguate the actor's plans is considerably reduced. The use of a reasoning mechanism that makes use of prior knowledge should improve the accuracy of these systems.

Another reason for using probabilistic approaches for plan recognition is to provide a theoretical basis for making default inferences to disambiguate plans. This is an important concern for both intended and keyhole recognition:

- Most approaches to plan recognition in dialogue systems have used heuristics for preferring one plan over another when there is more than one valid explanation for an observed action. These heuristics are usually based on properties of the dialogue in progress, such as the current context and focus of attention. However, it is possible that a speaker might intentionally fail to give explicit disambiguating information under the assumption that the listener will use common sense default knowledge to draw the necessary inferences.

- In his work on keyhole recognition, Kautz used a principle of minimizing the number of top-level goals in a hierarchy of plans to restrict the number of explanations his algorithm would consider.

In addition to the specific problems I have mentioned, both of these approaches will sometimes fail, in spite of their heuristics, to produce a unique explanation for the agent's actions. What they are lacking is an understanding of the fact that some plans are inherently *more likely* than others to be pursued. Probabilistic reasoning can provide such an understanding.

The three probabilistic plan recognition systems I have looked at make use of two different formalisms for reasoning under uncertainty: one uses Bayesian belief networks [11, 63], and the other two use Dempster-Shafer theory [76]. The first model [12, 13, 14] addresses the problem of story understanding using Bayesian belief networks to reason about the plans being pursued by characters in the stories. The second model [10] is concerned with incorporating default inferences into reasoning about the plans of a user of a natural language consultation system. Carberry uses the Dempster-Shafer theory of evidence to generate default rules for deriving conclusions from observations. The third model is an application of Dempster-Shafer theory to modeling user preferences in order to facilitate keyhole recognition in a computerized help system [6]. This work is notable in that it actually has a mechanism for acquiring the probabilistic information it needs from previous observations of the user's behavior.

Probabilistic approaches to plan recognition have the advantage that they make use of a well understood methodology for representing probabilistic relationships between goals and actions, and for abductive reasoning from observed actions to goals. They also provide a clear metric for prefering one explanation over another. However, the major drawback of these approaches is that evaluation of probabilistic networks is NP-hard. Even relatively small networks can take a long time to evaluate completely. In a real-time application in which the response time is very important, this time complexity is unacceptable. Developing more tractable algorithms for network evaluation is currently an active area of research and in the future may result in the possibility of practical probabilistic plan recognizers.

4.2 Recognizing plans in trauma management

4.2.1 Representation

Plan recognition requires a representation of both the *actions* that agents can perform in the domain, and the *goals* that they can serve. Because we already had a planning system that produced validated management plans [18] and an extensive knowledge base representing conclusions, goals, and actions in the domain, it seemed natural to use the knowledge base and representation of plans from TraumAID in the plan recognizer as well. Not only could TraumAID tell us what goals were possible explanations for the physician's orders, but it could also tell us what goals were more *likely* to be pursued in the current context, under the assumption that physicians are likely to pursue more urgent goals first, and other goals in conformance with the principles of trauma care encoded in TraumAID.

The importance of a shared knowledge representation for planning and plan recognition was first recognized by Wilensky [93] in the domain of story understanding. Having TraumaTIQ reuse TraumAID's knowledge is not only convenient, but also desirable because it facilitates comparison of the plans constructed by both subsystems – the plan generated by TraumAID's planner and the model of the physician's plan inferred by TraumaTIQ's plan recognizer. This allows the critiquing system to identify discrepancies between the two plans, such as different approaches to addressing a goal, which it might want to include in a critique. The advantages of using the same domain knowledge representation for both plan generation and critiquing has been discussed by Forslund [25], although he does not have an explicit representation of goals in his PPA system.

The disadvantage of reusing TraumAID's planning knowledge is that a knowledge base that is designed for generating valid plans may still lack explanations for what people actually do. We therefore have had to incorporate additional knowledge about goals and plans that, while irrelevant to TraumAID's generating "gold standard" plans, may be needed to understand actual patient management.

As I have outlined in Chapter 2, TraumAID's plans are composed of *goals*, *procedures*, and *actions*. The complete set of goals, procedures, and actions and the relationships between them can be represented as a *plan graph*, such as the one shown in Figure 4.1.

Goals correspond to the *abstract plans* in a hierarchical representation like the one

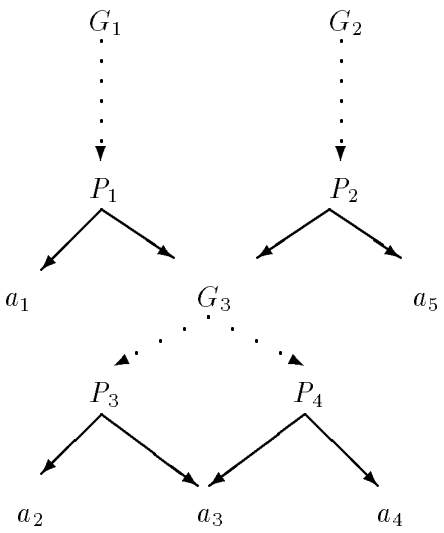

Figure 4.1: An example plan graph. Dotted arrows indicate disjunctive goal-procedure mappings, solid arrows indicate conjunctive procedure-action mappings

used in [42]. They represent high level objectives without specifying anything about how to achieve them on the action level. Each goal has associated with it a level of *urgency* and a *medical priority* derived from standard guidelines for trauma care, which are used as constraints in constructing the plan. The rule-based reasoner in TraumAID 2.0 is responsible for determining what goals are relevant to pursue, given the current state of the patient.

Procedures are somewhat analogous to *basic plans* in Kautz's representation. They are used to link abstract goals with the basic actions needed to address them. TraumAID associates each goal with a disjunctive list of procedures for addressing it. This association, represented by dotted arrows in Figure 4.1, is called a *goal-procedure mapping*. For example, goal G_3 can be addressed by either procedure P_3 *or* P_4. In each mapping, the procedures are ordered preferentially with respect to the goal. When planning to address a *set* of goals, TraumAID's planner selects one procedure for each goal from the goal-procedure mapping. Selection depends on both the local procedure preference rankings for each goal, and global considerations resulting from the need to address multiple goals concurrently.

A procedure comprises an ordered sequence of actions and/or sub-goals, stored in a *procedure-action mapping*, represented by solid arrows in Figure 4.1. Actions inherit their urgency and priority from the goals they are being used to address. An action always

inherits the urgency of the most urgent goal it is associated with. In addition to urgency, action definitions include:

- the approximate amount of time needed to perform the action,
- the sites in the hospital where the action can be done,
- the cost of the action,[2],
- contraindications to performing the action, and
- precedence constraints regarding other actions whose outcome could be influenced by the effects of the action, and so should be done before it if they also appear in the plan.

The use of sub-goals to address procedures allows TraumAID's planner to delay certain decisions about how to address its top level goals. For example, in Figure 4.1 G_3 is a subgoal of P_1. If TraumAID is planning to address G_1 by performing P_1 it can commit early on to the other actions in P_1, in this case a_1, while reasoning further to choose the optimal procedure for addressing G_3 based on the other currently relevant goals.

Actions and sub-goals can participate in more than one procedure at a time, and thus can be used in addressing more than one goal, as illustrated by sub-goal G_3, which plays a role in both procedures P_1 and P_2. This concept of *action overloading* is exploited by TraumAID's planner to produce globally efficient plans.

In the rest of this chapter, the words *goal*, *procedure*, and *action* will refer to the concepts as I have just described them.

4.3 Plan Recognition with Diagnostic Actions

Diagnostic actions are actions that are taken to acquire knowledge about the world. They can be linguistic, such as asking a question, or physical, such as listening to a patient's breathing or taking an X-ray. Some diagnostic actions are sensing actions, and have an immediate result. Others require a response from an external source, and may involve

[2]The cost is a subjective disutility of experiencing the action. The way in which these costs were determined is discussed in Chapter 5.

a period of waiting between performing the diagnostic action and receiving the desired information. In this section I will describe how knowledge about diagnostic goals and actions used in the management of multiple trauma can be exploited to help recognize the physician's plans [33].

Consider an agent as in [8] who *reacts* to the world by forming goals, who *reasons* to decide which goals to adopt and attempt to satisfy in which order, and who then *acts* on his or her decisions. Such an agent may carry out diagnostic actions for one or more of the following reasons:

- *Goal Selection.* The agent may desire information in order to decide whether or not to adopt a goal suggested by the environment. For example, what it has seen of the world may suggest to the agent that it adopt a goal of having lots of money. The agent may then desire to know any down sides to attempting and possibly achieving the goal (e.g., increased attention from the IRS), so that it can weigh its *expected benefits* against its *expected costs*.

- *Goal Refinement.* The agent may desire information in order to decide how to refine an abstract goal like "return a patient to health" (What's wrong with the patient?) or "live a good life" (What are the elements of a good life?).

- *Goal Decomposition.* An agent may desire information in order to decide which of several ways to satisfy a specific goal like "travel to Washington" (What's the cost to fly, drive, or take the train? What's the overall transit time in each case?) or "diagnose a possible kidney injury" (Is the patient in shock? If so, there's no time to take the patient to radiology for an IVP). Notice in the latter case, that the goal itself may be diagnostic.

- *Knowledge Preconditions.* An agent may need information in order to carry out a particular action like "phone up Martha" (What's her area code?) or "give antibiotics" (Which, if any, is the patient allergic to?)

In work to date on cooperative question-answering, the purpose of diagnostic actions has mainly been for *goal decomposition* [44, 69] and *knowledge preconditions* [58, 59]. On the other hand, in the initial definitive management of multiple trauma, physicians carry

out diagnostic actions primarily for purposes of *goal selection* and *goal refinement*. To enable plan recognition techniques to be developed for these latter purposes as well, it is essential to consider the notion of *diagnostic strategy*.

An agent lacking knowledge of the current situation may not even know what it needs to know in order to satisfy its goals. A diagnostic strategy will direct the agent to gather particular kinds of information – i.e., to perform particular kinds of diagnostic actions. After gathering one piece of information, an agent who still lacks sufficient information may stick with the same strategy or mix strategies according to some algorithm or set of heuristics. A sequence of strategically-motivated actions is desirable if it minimizes the (agent's) cost to acquire the relevant information, where cost may be in terms of money, time, risk of mortality or morbidity, or any combination of these or other costs.

These diagnostic strategies are similar to the problem-solving and plan-exploration meta-plans represented in [44, 69], but we are concerned with a wider range of problem solving activities, such as goal selection and goal refinement. Appealing to diagnostic strategies requires that a plan recognition system understand the knowledge-changing capabilities of actions and relationships between knowledge and action.

Several medical expert systems make use, or have made use, of diagnostic strategies in their reasoning. For example, INTERNIST [66] was developed as an expert diagnostic system in the area of general internal medicine. When INTERNIST is given an initial set of salient disease manifestations, it forms a list of plausible disease hypotheses (a *differential diagnosis*) and then bases its request for additional information on one of three diagnostic strategies or "modes", depending on how many diseases, if any, are "close" to the highest scoring hypothesis.

In the initial definitive management of multiple trauma, the strategies used are determined by the urgency of addressing potentially life-threatening injuries, the high cost of various procedures in terms of both risk and money, and the ability to appeal to surgical procedures for both definitive injury investigation and repair. Because in multiple trauma the presence of multiple injuries means that practitioners need to integrate diagnostic and therapeutic activity into a single management plan, I will use the term *management strategy* rather than *diagnostic strategy* when talking about goal selection in trauma. Such strategies are embodied in the reasoning and planning knowledge used by TraumAID 2.0

[75].

Since trauma management requires the concurrent consideration of both diagnostic and therapeutic issues, TraumAID's planner treats diagnostic goals and therapeutic goals as the same type of object [75]. Both are concluded relevant by the reasoner as a result of the appropriate evidence being known, and both are associated with procedures and actions in the same way. In the plans generated by TraumAID, diagnostic and therapeutic procedures can be interleaved in any order to improve the overall efficiency of the plan. TraumaTIQ is therefore able to treat diagnostic actions the same way as therapeutic actions in recognizing their underlying goals: each diagnostic action may be done to address a small number of goals, and the task of plan recognition upon observing one of these actions involves selecting one or more of the associated goals to explain the action.

An understanding of typical management strategies can enhance the plan recognizer's ability to infer goals underlying both diagnostic and therapeutic actions by linking related goals which may become relevant in similar situations. For example, one strategy employed in TraumAID is to filter out potential diagnoses using quick "bedside questions" before going on to more definitive diagnostic testing. These initial questions are used for the purpose of *goal refinement*. They are usually not accurate enough to justify drawing any definite conclusions, but they can be valuable in pointing the physician in the right direction and eliminating unnecessary diagnostic tests from the plan. If the physician is observed to order a test that may be used to address a diagnostic goal that is not currently motivated but may *become* motivated as a result of a currently pending bedside question, TraumaTIQ can infer that the physician is pursuing that diagnostic goal, albeit prematurely.

4.3.1 Characteristics of the plan recognition problem

As Goodman and Litman [34] 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.

As we have seen in Section 4.1, previous approaches to plan recognition have relied on

a number of assumptions about the user, the domain, and the system's knowledge, some of which have been enumerated by London [48]:

Closed world assumption The system has complete knowledge of the plans and actions that may be performed by the agent.

Plan correctness The agent always reasons with correct beliefs and acts according to a coherent, well-formed plan.

Unified goal and plan The agent pursues a single top-level goal at a time.

Intended recognition The agent acts in such a way as to help the observer infer her underlying plans.

No real-time requirement There is no bound on the amount of time that may pass before the system responds to the agent's actions.

As London points out, most human activity violates all of these assumptions in various ways, and making use of such assumptions in plan recognition can seriously restrict the available uses of the resulting system. In the case of TraumaTIQ, we need to drop all of them.

The closed world assumption is not valid for TraumaTIQ because the domain of TraumAID only covers a subset of plans that physicians may carry out in a hospital. Even when the knowledge base is extended beyond penetrating injury to the chest and abdomen, there are still many plans that may be followed in the context of a trauma case that will not be anticipated ahead of time.

TraumaTIQ cannot assume that the physician's plans are always correct. The system must have some guidelines to interpret orders that are motivated by misconceptions or errors in judgment.

While TraumAID and the physician may be pursuing a single top-level goal (restore the patient to health) this may involve many sub-goals, some of which address different injuries possibly resulting from different wounds. But even 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.

The plan recognition problem for TraumaTIQ is an instance of *keyhole*, rather than *intended* recognition, meaning that the physician does not act with the intention of having the system understand her actions. Its "keyhole" observations correspond to entries made by the recording nurse:

- *actions* that have been performed,
- the results of those actions in terms of both changes in and new information gained about the patient's condition, 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.

The urgency inherent in trauma management means that there is limited time available to respond to physicians' actions. It is therefore very important to limit the amount of computation necessary for the plan recognition algorithm.

Another feature of the plan recognition problem for TraumaTIQ is that plans must be recognized incrementally, *during* patient management. Therefore, the plan recognition algorithm must take 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 TraumaTIQ has access to a great deal of relevant information about the situation through the interface to TraumAID, into which the scribe nurse enters data as it becomes available. 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.

Another advantage for plan recognition in TraumaTIQ is that the physician will have training and experience in trauma management. This licenses two assumptions that can

be used to guide plan recognition:

- 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 sometimes proceed more rapidly, acting on the basis of evidence that for TraumAID's reasoner would be insufficient.

The first assumption motivates a policy of giving the physician "the benefit of the doubt": if an ordered action can be explained in terms of TraumAID's current goal set, the physician will be assumed to be pursuing the explanatory goal(s). An ordered action can be explained if it appears in TraumAID's plan for addressing a goal in the goal set, *or* if TraumAID has chosen a different action to address this goal. A related aspect of this policy is to ascribe goals that are *partially* supported according to TraumAID to the physician in preference to goals that are not supported at all. The definition of partial support that we are using here is discussed later in this chapter.

In general, this policy biases the plan recognizer to ascribe fewer incorrect goals to the physician. This bias is justified in our application because the main purpose is to identify when an action would compromise patient care. If it wouldn't, it is not important for the critique to identify why it is being done.

The second assumption allows the plan recognizer to interpret actions that could be justified by more evidence. Identifying when physician orders may be motivated by a goal that is partially, but not yet completely supported by the evidence requires a representation of the strategic relationship between goals.

4.3.2 Using context to interpret actions

Several researchers have pointed out the advantages of using contextual knowledge and basic domain principles to guide the search for an explanatory plan [40, 38, 49, 48]. The basic idea behind these approaches to plan recognition is that the plan recognizer can use its knowledge of what actions are appropriate to take in the current situation to reduce ambiguities in interpreting observed actions.

Huff and Lesser [38] describe GRAPPLE, an incremental plan recognizer for intelligent assistance that uses contextual knowledge to simplify computation in choosing the most likely explanations for observed actions. GRAPPLE uses a truth maintenance system (TMS) to reason about the hidden state of the system being worked on. When an action is observed, candidate interpretations for that action are evaluated in terms of whether their preconditions are satisfied, their constraints are met, and that the goals they address are not already achieved or planned for in some other way. They do not, however, consider the *relevance* of competing interpretations in the current context.

London's IMAGE student modeller for the Guidon2 tutor [49, 48] uses the reasoning of the underlying knowledge-based system, NEOMYCIN [16], to form predictions about the potential actions of the student that would be relevant relevant, given the current knowledge of the situation. These predictions serve as a context in which to interpret the student's observed actions.

Hill and Johnson's "situated plan attribution" [40] is an approach to plan recognition for tutoring which uses knowledge about the task the student is trying to perform to interpret and correct inappropriate actions. Their REACT system tracks the student's actions and interprets them in terms of the currently relevant plans for accomplishing the task. If an action cannot be matched with any relevant plan, an *impasse* is detected, and further evaluation is used to resolve the impasse, either by recognizing it as a valid deviation from the plan, or by informing the student of the error.

The student modelling systems presented in IMAGE and REACT rely on the assumption that students are likely to act in a way that an expert would see as being motivated by the current situation. In tutoring applications, this is a rather weak assumption, since the student is by definition not an expert in the task domain. In the IMAGE domain, students' actions directly agree with what NEOMYCIN would do only 50% of the time. By incorporating domain specific knowledge about likely student errors, IMAGE is able to anticipate 58% of observations. In contrast, in a decision-support application, the user *is* expected to be a domain expert, and so the assumption of user competence is much more strongly justified. In experiments with TraumaTIQ which will be discussed further in Chapter 8, 60% of physician actions are in agreement with TraumAID's current plan. Including actions that can be considered *potentially relevant* given a small amount of additional knowledge,

this figure jumps to 75%.

Plan recognition in TraumaTIQ takes advantage of three types of contextual information that influence the likelihood that the physician is pursuing a certain goal:

The likely goals to be pursued in the current situation Given TraumAID's current information about the state of the patient, TraumaTIQ is able to make certain inferences about what goals are more or less *relevant* to pursue.

The physician's actions The more evidence TraumaTIQ has that the physician is performing a procedure, the more likely it is that she is actually performing it.

The likelihood of procedures being used to pursue a goal While it may be possible to pursue a goal in a number of ways, some of them may be quite uncommon. This is reflected in the preference ordering for procedures in TraumAID's knowledge base.

4.3.3 The Plan Recognition algorithm

The task of the plan recognizer is to build incrementally a model of the physician's plan based on the actions being ordered. Following the assumptions given above, TraumaTIQ's plan recognizer prefers to explain the physician's actions in terms of goals (and procedures) that TraumAID currently considers *relevant* to the case.

A formal description of the plan recognition algorithm appears in Figure 4.2. Basically, it works as follows: It first enumerates the set of possible *explanations* for all actions that have been ordered. Each explanation consists of a path in the plan graph from the ordered action to a *procedure* in which the action plays a part, back to a top level *goal*. The path may pass through a series of sub-goals and procedures before reaching a top level goal. Since the same goal may be addressed by more than one procedure, it is possible for an action to be explained by a goal in the context of two different procedures. For example, in Figure 4.1, action a_3 is explained by goal G_3 through both procedures P_3 and P_4.

The possible explanations are evaluated in two phases. The first phase considers the *goals* in the explanations. These are sorted according to their *relevance* in the current situation. The plan recognizer categorizes potential explanatory goals according to a 4-level scale of relevance:

1. Relevant goals: goals that are in TraumAID's set of goals to be pursued.

2. Potentially relevant goals: goals that are part of a currently active *diagnostic strategy*. As described in Chapter 2, diagnostic strategies are represented implicitly in TraumAID's knowledge base. They comprise chains of goals each of which, given the appropriate result, leads to the formation of the next goal in the strategy. So, for example, if the goal of diagnosing a fractured rib is currently relevant, then the goal of treating a fractured rib is potentially relevant, depending on the result of the diagnostic test.

3. Previously relevant goals: goals that were once relevant but are no longer relevant, either because they have been addressed or because some additional evidence has ruled them out.

4. Irrelevant goals: all other goals are classified as *irrelevant*.

The assumption underlying this phase of plan recognition is that the higher a goal is on this scale, the more likely the physician is to be pursuing it. The most relevant ones are selected as candidate explanations for the orders.

The likelihood that a goal is being pursued depends not just on its relevance, but also on the likelihood that the ordered actions would be used to address it. Therefore, goals that are potentially or previously relevant are not accepted as explanations unless the ordered actions play a role in the *most preferred* procedure for addressing those goals.

As others have pointed out [89], depending on the reason for doing plan recognition, it is not always necessary to infer a unique goal or goals for every action. For the purpose of critiquing, we do not want to spend time interpreting actions that are clearly incorrect, since they are harder to understand and will be mentioned as being unmotivated in the critique regardless of the physician's reason for doing them. Therefore, if there is more than one possible explanatory goal, none of which is relevant, the algorithm does not try to disambiguate the explanation further and the process halts here. Otherwise, the highest ranking (most relevant) non-empty subset of explanatory goals is selected to be evaluated in phase two.

The second phase considers the procedures in the remaining explanations. These are

evaluated according to how strongly the physician's other actions/orders provide additional evidence for them. The more actions in the procedure have been ordered, the more evidence there is in support of the explanation. For simplicity, the procedures are actually sorted according to a four-level scale of evidence:

1. Completed procedures: procedures for which all the actions have been ordered by the physician.

2. Partially completed procedures: procedures for which some of the actions have been ordered.

3. Relevant procedures: procedures that are currently in TraumAID's plan. This means that if an action could address a goal by playing a role in two different procedures, the procedure that TraumAID has selected in its plan is preferred as the explanation for the physician's action.

4. All other procedures.

All procedures in the highest non-empty category are accepted as explanations for the action.

Finally, the explanations with the most relevant top-level goals and the greatest amount of observed evidence are ascribed to the physician and incorporated into TraumaTIQ's model of the physician's plan. Incorporating a new explanation into the plan involves adding new procedures and goals if they are not already present, and adding links between items that are not already connected.

Note that there may be more than one explanation for a given action, as long as the explanatory goals are equally relevant, and the procedures have the same amount of observed evidence supporting them. For example, in Figure 4.1 both G_1 and G_2 might be accepted as explanatory goals for the action a_3, provided that both goals are in the same category of relevance, and are not irrelevant.

1. For each action α ordered, TraumaTIQ's plan recognizer extracts from TraumAID's knowledge base a set of *explanatory procedure-goal chains*, PG_α, that could explain the presence of that action:

$$PG_\alpha = \{\langle P \ldots G\rangle_1, \ldots, \langle P \ldots G\rangle_n\}$$

   where P is a procedure containing α in its decomposition, and $\langle P \ldots G\rangle_i$ is a backward path through the plan graph ending with the goal G.

2. Now consider the set $\Gamma = \{G_i\}$ where G_i is the top level goal ending $\langle P \ldots G\rangle_i$. In rank order, Γ consists of Γ_1 the relevant goals, Γ_2 the potentially relevant goals, Γ_3 the previously relevant goals, and Γ_5 all other goals. Let $\Gamma' = \{G_j\}$ be the highest ranking non-empty subset of Γ. If Γ' is the set of irrelevant goals, halt here and add α to the plan with no explanatory procedure-goal chains.

3. Let $\mathcal{P} = \{P_j\}$ where P_j is the procedure that is the child of G_j in PG_α. In rank order, $\mathcal{P}$ consists of: $\mathcal{P}_1$, procedures for which all the actions have been ordered, $\mathcal{P}_2$, procedures for which some of the actions have been ordered, $\mathcal{P}_3$, procedures that are currently in TraumAID's plan, and $\mathcal{P}_4$, all other procedures. Let $\mathcal{P}'$ be the highest ranking non-empty subset of $\mathcal{P}$.

4. Select the paths $PG' \subseteq PG$ such that PG' contains all paths ending with goals in Γ' with children in $\mathcal{P}'$.

5. The paths in PG' are then incorporated into TraumaTIQ's model of the physician's plan, connected to the action α.

Figure 4.2: The plan recognition algorithm

4.3.4 Complexity of the plan recognition algorithm

A serious criticism of previous approaches to plan recognition is that they are computationally intractable. For example, Goodman and Litman explore the design of plan recognition algorithms through CHECS, an implementation of Kautz's algorithm in the domain of chemical process design [34]. They remark that the system experiences a computational explosion during search for explanatory plans, and thus "although CHECS eventually got the correct answer, it was slow getting there" ([34], page 104). Similarly, the computational resources required by the probabilistic plan recognizer implemented by Charniak and Goldman [14] grow exponentially with the number of actions in the input, causing the system to run unacceptably slowly.

As I have pointed out previously, in a time-critical domain like trauma management it is

essential for TraumaTIQ to respond quickly. The complexity of the algorithm is not really a problem in the current implementation of TraumaTIQ because of the limited domain for which the knowledge base is designed. To get an idea of the size of the problems we are dealing with, here are some relevant statistics:

- In the database of 97 actual cases, there is an average of 6 actions in each management plan. The plan recognizer builds an explanatory plan for every order that has not yet been executed, so even if *all* the actions are ordered before any of them are carried out, the plan recognition algorithm will on average have to explain no more than 6 orders at one time.

- The branching factor is generally small. Of the 105 diagnostic and therapeutic actions in TraumAID's knowledge base, 76 of them participate in only one procedure, 15 in two procedures, 4 in three procedures, 2 in 5 procedures, and 1 each in 13 and 19 procedures.

  Of the 95 procedures in TraumAID's knowledge base, 62 of them can be used to address only one goal, 14 to address two goals, 3 to address three goals, 5 to address 4 goals, 2 to address 6 goals, 1 to address 7 goals and 2 to address 9 goals.

So in general, the plans TraumaTIQ has to recognize are quite small, and the branching factor of plans in TraumAID's knowledge base is small as well.

To demonstrate how fast the implementation actually is in practice: TraumaTIQ's plan recognizer, implemented in Lucid Common Lisp and compiled on a Sun 4 processed 584 actions in an average of 0.023 cpu seconds per action.

The problem arises when we consider extending the system to cover other areas of the body and/or blunt injury. This will result in increasing the number of procedures and goals that might explain an action in the knowledge base. To allow for the growth of the system, it is important that the plan recognition algorithm scale up efficiently.

As Rymon [75] points out, plan recognition can be formalized as a set-covering problem in which two sets of observations, symptoms and actions, are mapped onto a set of goals which covers both of them (every symptom motivates some goal and every action is motivated by some goal in the covering set – see Figure 4.3). The covering set is optimized according to some cost function (e.g. in Kautz's plan recognition formulation the

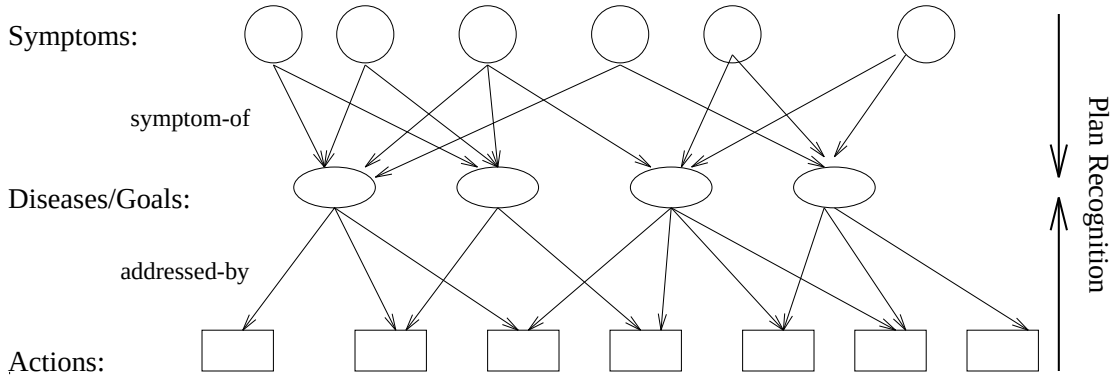

Figure 4.3: Plan Recognition as a set covering problem

cost function is minimization of the goal set). Since the set covering problem in general is NP-hard, so is this formalization of plan recognition.

Actually, the relationship between symptoms and goals in plan recognition is not the same as the relationship between goals and actions: it is important for the inferred goal set to be *complete* in its coverage of all the observed actions, but it is not necessary that the inferred goals cover all of the observed symptoms – we do not assume that the agent is addressing all of the goals that are motivated by her environment.

In general, any plan recognition algorithm that considers all possible combinations of explanatory goals for the observed actions is going to grow exponentially with the number of actions. The algorithm I have presented here avoids the need for an exponential search by grouping the potential explanations according to *relevance* and then greedily accepting all the explanations in the most relevant group. One way to look at this is that rather than trying to optimize the covering goal set according to a cost function, we simply choose to maximize the number of relevant goals in the covering set.

In doing this, for each ordered action α, the algorithm only has to consider $|\Gamma|$ goals, where Γ is the set of possible explanatory goals for α, and $\sum_{|\Gamma'|} |\mathcal{P}_{\Gamma_j}|$ procedures, where Γ' is the most relevant non-empty subset of Γ, and $\mathcal{P}_{\Gamma_j}$ are the procedures linked to each goal Γ_j in Γ'. For each procedure, it has to look at $|\mathcal{A}_\mathcal{P}|$ actions in the procedure, and compare them with at most all of the actions that have been ordered. So the total cost of inferring a plan for a set of orders, $\mathcal{A}$, is at most

$$|\mathcal{A}| * (|\Gamma| + (\sum_{|\Gamma'|} |\mathcal{P}_{\Gamma_j}| * |\mathcal{A}_{\mathcal{P}_j}| * |\mathcal{A}|))$$

. Thus, this algorithm is polynomial in the number of ordered actions, and linear in the number of possible goals per action, the number of goals in the most relevant goals set, and the number of possible procedures per action.

In this situated view of plans, in which we interpret actions using the context in which the agent is acting, an optimal explanation would be one that both maximized the number of relevant goals and minimized the number of irrelevant goals. TraumaTIQ's greedy selection of relevant goals achieves the former – the selection of *all* possible relevant goals is justified by the observation that overloading actions to address multiple goals is advantageous, and because we have chosen to give physicians the benefit of the doubt when they seem to be acting in accordance with TraumAID's goals. On the other hand, the algorithm does not generate a minimal global explanation of the set of unmotivated ordered actions. This has the result that it produces less refined explanations of these actions. For example, if two unmotivated actions – giving antibiotics and immobilizing the patient – are ordered, TraumaTIQ will not be able to ascribe a unique goal to either action even though there is only one goal – treating a compound fracture of the vertebrae – that calls for both of them.

This weakness in interpreting unmotivated actions reflects a design decision not to devote computational resources to understanding erroneous behavior. Instead, TraumaTIQ simply responds to these actions by commenting that none of the goals that might possibly motivate the action are currently relevant according to its information about the patient. As long as one of these possible motivations is the one the physician is pursuing, she should be able to understand the comment and either rescind the order or provide the missing information to the system.

This greedy selection of relevant covering goals works only because we have an idea of which goals are relevant given the patient's symptoms, and because we make the assumption that an experienced physician is likely to pursue relevant goals. When the observed actions cannot be interpreted as addressing any relevant or potentially relevant goal, the algorithm breaks down. The restriction to experienced users could be loosened if the system were able to model the relationship between different situations and the typical goals of less experienced users. Rather than accept all goals that are relevant according to a *correct* model of the domain, we might accept the goals that the user we are working with

would consider relevant.

In general then, this approach to plan recognition is applicable in any domain in which the user's typical behavior can be modeled in context, and in which it is less important to understand irrelevant or atypical behavior. For example, if we knew that medical students usually try to treat a hemothorax by ordering a chest tube when they hear muffled heart sounds, even though the appropriate goal is to rule out a pericardial tamponade by getting a needle aspiration of the chest, we would be able to understand the order of a chest tube in that situation as being related to the muffled heart sounds. But if we had never seen a student diagnose a hemothorax on the basis of muffled heart sounds, we could simply respond that: "Getting a chest tube seems unmotivated. Please provide more information regarding this order."

4.4 Evaluation of the Plan Recognition Algorithm

To evaluate its performance, I have applied the plan recognition algorithm to the management plans from 97 actual trauma cases from the Medical College of Pennsylvania. These cases have been abstracted for the purpose of comparing them to plans generated by TraumAID [18], and only contain actions that are in TraumAID's knowledge base. The plan recognition algorithm is designed so that it always identifies a goal or goals underlying actions that are in TraumAID's plan at the time they are ordered. The difficult actions to interpret are the ones that do not appear in TraumAID's plan, since they cannot easily be explained in terms of TraumAID's current goals.

Out of 584 actions, 234 of them were not also part of TraumAID's plan at the time that they were ordered. Of these 234, 15 of them could be explained by a goal that was currently in TraumAID's relevant goal set. Of the remaining 219, 69 could be explained by a goal that was considered to be potentially relevant, given TraumAID's current knowledge about the state of the patient. The plan recognizer failed to explain the remaining 148 actions in terms of relevant or potentially relevant goals.

Part of the reason for these plan recognition failures is that the knowledge base designed for TraumAID's planner is insufficient for the needs of plan recognition. First, many of the actions that TraumaTIQ fails to infer a goal for are broad diagnostic tests that can

be used to look for a number of conditions, and the physician may not actually have a specific goal in mind when ordering them. To understand physicians' plans in such cases it is necessary to have a representation of abstract goals that is not currently available in TraumAID 2.0. Since the knowledge base was implemented in support of plan *generation* rather than plan *recognition*, only goals that could be directly operationalized as actions were included.

Second, some goals that physicians may pursue in these cases are not included in TraumAID's knowledge base because its designers opted not to pursue these goals under any circumstances relevant to the current domain of the system. To have a complete plan recognition system, it is necessary to include such goals in the knowledge base. In Chapter 7, I will discuss these and other additions to TraumAID's knowledge base that are necessary to support plan recognition.

Another weakness in the plan recognition algorithm comes from the method for comparing the amount of observed evidence for different explanatory procedures. To compare hypotheses, I have grouped procedures that have been totally ordered in one category, and procedures that have been partially ordered in another. This method is not ideal, however, because some actions are more central or important in a procedure than others. For example, the procedure for performing a peritoneal lavage involves inserting a Foley catheter and then lavaging the abdominal cavity. The first of these actions is preparatory, while the second is the main part of the procedure.

Ideally, we would like to be able to determine how often the plan recognition algorithm infers the *correct* goal(s) for an action. This evaluation is not possible, however, without data that includes both the actions performed and the underlying goals motivating them. Unfortunately, this information is not available in the abstracted cases we have available to us so far. Therefore, an evaluation on this level cannot yet be done.[3] During the planned prospective evaluation of TraumAID, physicians will be asked in a debriefing questionairre to give their reasons for doing certain actions. This information can then be used to evaluate the accuracy of the plan recognizer.

[3]To my knowledge, no plan recognition system has yet been evaluated on this level.

4.4.1 Relevance and a possible probabilistic approach

This algorithm relies on the identification of partially relevant goals to discriminate between more or less likely explanations for ordered actions. Consequently, the inability to make fine-grained distinctions between degrees of relevance of goals limits its accuracy. Goals are classified as *relevant* as long as TraumAID has enough evidence to justify pursuing them, but some relevant goals may well be *more* relevant than others and still other goals may be *almost* relevant.

I have attempted to capture the concept of almost relevant goals using the concept of potential relevance, which says that if a goal has a reasonable chance of becoming relevant in the future, then it is not completely irrelevant in the present. However, a goal may also be considered almost relevant if some, but not all, of the evidence needed to make it relevant is now available. I have not been able to model this in my system because it is not possible to tell how strongly a conclusion is supported simply on the basis of the number of items of evidence associated with it. The evidence supporting conclusions does so with varying degrees of strength, and without any indication of the relative influences of different pieces of evidence, which is not available in TraumAID's rule base, we cannot determine the proportion of support for a conclusion that is not fully supported.

A probabilistic framework would make these distinctions possible. While it is difficult to accurately model the likely behavior of individual physicians, this problem can be greatly simplified by assuming that the probability that a goal *will* be pursued is close to the probability that the goal *should* be pursued, as measured by the expert system, perhaps in terms of the expected utility of pursuing the goal. This expected utility can be thought of as an approximation of the prior probability that a goal will be pursued, given the current state of the case.

Given an expert system that is capable of producing a numerical value for the probability that a goal *should* be pursued, a probabilistic model of plan recognition in the trauma domain could be developed which would be able to make much more fine-grained distinctions between hypothesized explanatory goals than the current algorithm is capable of. For example, consider a Bayes net representation of plans such as the one presented in [14]. Charniak and Goldman used a uniform distribution over all events in their universe

to initialize the prior probabilities in their networks, so in reading a story it is equally likely *a priori* that the word go will refer to going to the liquor store or going to the movies.

In contrast, in the model I envision, the prior probabilities for goals could be taken from the expected goal utilities, and used to condition the inferences resulting from future observations. So, if the expected utility of pursuing a diagnosis of hemothorax is higher than the expected utility of pursuing a diagnosis of pneumothorax because the clinical indications of hemothorax are stronger, then the prior probability that the physician will try to rule out a hemothorax will be higher than that for a pneumothorax. Consequently, after observing the physician ordering a chest x-ray, the system could infer that the goal of ruling out a hemothorax is the most likely explanation for that action, given that both ruling out hemothorax and ruling out pneumothorax can be done by getting a chest x-ray.

I have begun exploring the possibility of such a model using the IDEAL system for representing and solving Bayesian belief networks [83]. However, as mentioned earlier, the complexity of network evaluation currently poses a problem for the practical application of this approach.

Chapter 5

Outcome-Driven Plan Evaluation

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 [18, 75]). 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.*

Plan recognition allows TraumaTIQ to develop a global picture of what the physician is doing based on the actions she has ordered. But it is not enough for a critiquing system to understand what is being done in a given situation: it must also be able to identify potential errors in the plan and determine how to respond to the user. This is the role of plan evaluation, which uses both the inferred representation of the physician's plan, together with TraumAID's current understanding of the case to generate a set of *critique comments.*

5.1 Differential vs. Analytical plan evaluation

In general, plan evaluation can be done using a *differential* or an *analytical* approach [23, 22]. The former method compares the user's plan to a *target* 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 target plan that can be assumed to be a broadly acceptable (if not optimal) 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 detects possible interactions between goals, and find the most efficient way to address that particular combination of goals (see Figure 2.5). A differential evaluation of a user's plan can determine when the user is not reacting to potential interactions between goals simply by looking at the interactions that were identified in forming the target plan.

On the other hand, the analytical approach has the advantage that it does not depend on a single solution as the basis for critiquing the user. 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

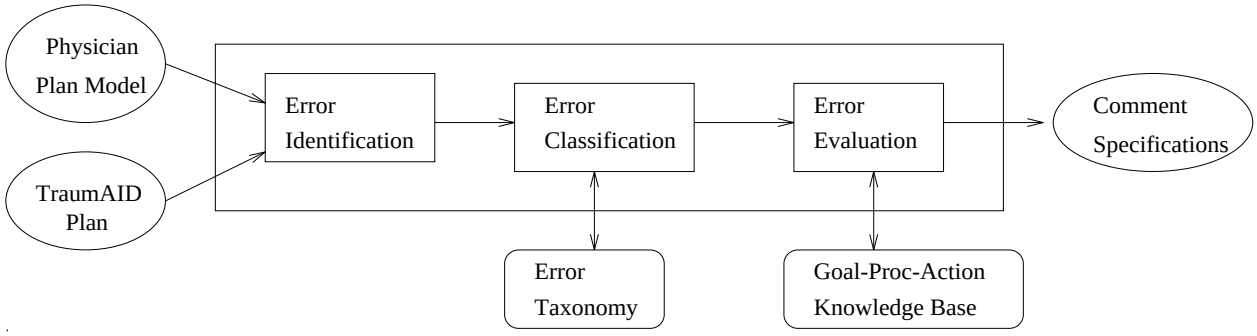

Figure 5.1: The plan evaluator

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, such as the JANUS system [22], might include a rule that the stove should be no less than five feet from the sink. If this rule is violated in a proposed design, the system can cite it as a reason for not accepting the design.

The approach to plan evaluation that I have developed 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.

The plan evaluator is primarily differential. It uses the plan generated by TraumAID's planner as the target plan, and compares it with the plan inferred by the plan recognizer. However, by augmenting the planner to record the decisions made during the planning process, I have made it possible to explain not only why TraumAID's plan includes the actions it does, but also why certain actions are *not* in the plan.

I have also made the plan evaluator more tolerant of minor deviations from the target plan than a purely differential analysis would allow. Using knowledge about the magnitude of different types of errors, it filters the output so that only non-trivial errors are critiqued. This filtering process also results in a classification of the remaining errors as either critical or non-critical, which is later used to determine the final organization and form of the output.

Plan evaluation is done in three phases, as shown in Figure 5.1. First, the physician's plan is compared to TraumAID's plan and all discrepancies between the two are identified. Next, the discrepancies are classified using a taxonomy of error types. Finally, each error that has been classified is evaluated on a three-level scale of significance using information from TraumAID's knowledge about goals, procedures and actions to determine the final contents of the critique.

5.2 Identifying and classifying human errors

Determining what kinds of errors people make and when they are likely to make them has been a primary concern of work in the field of human-computer interaction. Of particular relevance to this thesis is work on the development of decision-support systems for process control environments [37, 71, 73]. These models are intended to identify errors committed by operators of complex technological systems, and to respond such a way as to minimize the effect of the errors. Work in this area began by drawing from evidence from cognitive psychology regarding the evaluation and classification of human errors. The idea was to develop an operational definition of different types of errors based on the understanding of what causes those errors to occur. In the next section I will discuss the psychological theory of biases in judgment and decision making, and describe an approach to critiquing that makes use of this theory [80].

5.2.1 Recognizing the causes of error

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 [41]. 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 studies in cognitive psychology, such as those presented in [41]. It has often been observed that people make

use of heuristic rules when forming judgments based on uncertain information. 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, it has been claimed that these heuristics can lead to systematic, predictable biases [87]. 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 [17, 79].

There are three types of reasoning that have been associated with this heuristic process, and in which biases have been observed: *probabilistic reasoning*, *causal reasoning* and *logical deduction*. One heuristic used in probabilistic reasoning is referred to as the *availability* heuristic, which 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 highly salient evidence.

Errors in causal reasoning include a phenomenon that is referred to as the *fundamental attribution error*, in which observers tend to attribute the actions of others to personal disposition, while the actors themselves attribute their actions more strongly to situational factors.

An example of a typical error in logical deduction is the *confirmation bias*. This is a phenomenon in which people tend to test only instances which support their current hypothesis, and not those that would refute it. This strategy allows hypotheses to be confirmed more quickly, but may result in erroneous conclusions if the hypotheses are not correct.

In Chapter 3 I introduced an approach to critiquing that makes explicit use of knowledge about these types of judgment biases [79, 80, 82]. Silverman's approach is to develop operational rules for identifying errors resulting from the underlying catalog of biases. For example, a rule for the availability bias might say that if a person is observed to be using only easily available knowledge or ignoring knowledge that is not easily available, then the availability bias is likely to have occurred. The resulting critique can explain the negative effect of this bias and propose a different approach.

Silverman's approach is interesting in that it attempts to identify and correct the underlying *causes* of human error. However, it relies on the correctness of the particular theory of cognition and action put forth by Kahneman, Tversky and others. Unfortunately, it is not clear that this theory has been validated enough to be used in this way [28]. The experimental evidence for many of the biases and errors reported in the literature has been contradicted or has failed to be replicated in different contexts. It may be the case that certain biases only appear in very specific (perhaps artificial) situations, the characteristics of which have not yet been sufficiently explored. In addition, in the case of some of the biases described in the literature, it has been argued that the definition of normative or "correct" behavior is not well defined, and that in fact the observed behavior cannot really be called erroneous.

Furthermore, as Silverman himself points out, even if a proven cognitive theory did exist, defining the observable manifestations of each bias in any given domain would be nearly impossible. The rules he gives for recognizing occurrences of errors due to cognitive bias are much too high-level to apply in any real situation, and refining them further is a costly and highly domain-specific task. For example, in looking for the availability bias, how can we define what information is easily available? The availability of information may depend on user-, domain- and situation-specific factors, such as the memory of the user, the possible information sources, and the physical location of those sources. Any rule for identifying the availability bias would only be valid in a very small range of situations.

A final objection to this approach is that it seems to be at odds with the spirit of critiquing. The purpose of a critiquing system is to provide assistance to a trained individual during the performance of a task. As such, the system should be able to identify when a specific error is being made and to correct or avoid that error. Silverman's approach of

recognizing when particular cognitive biases are likely to affect the user's behavior looks much more like a tutoring approach, aimed at reducing the occurrence of those biases in the long run. While combining critiquing and tutoring capabilities may be a good idea in some applications, it is important to recognize the difference between the two. Addressing the underlying causes of errors is both more complex and more time consuming than addressing their observable manifestations, and so would not be desirable in some applications of critiquing, particularly those that are designed for time-critical tasks. In the next section, I will describe an approach that focuses on identifying the observable manifestations of errors rather than their underlying causes.

5.2.2 Recognizing the manifestations of error

In his work on identifying human errors in process control environments, Hollnagel [36, 37] has made an important distinction between the underlying *cause* or *genotype* of an error, and the observable *manifestation* or *phenotype* of the error. Silverman is concerned primarily with the former, and has defined categories of observable erroneous actions based on the underlying theory of cognitive bias described by Kahneman and Tversky. Other work on classifying human error [71] has also been primarily concerned with the causes of error rather than the errors themselves.

But, as Hollnagel points out, it is often a mistake to mix the classification of observable phenomena with the interpretation of their causes. For example, an agent who fails to carry out an action will often be said to have "forgotten" the action, when actually the observable manifestation of the error can only be seen as an omission, which may or may not be caused by forgetting. Furthermore, depending on the purpose of identifying the error, the cause of the omission may be irrelevant. Once the agent is "reminded" to do the action, the reason he had omitted it may not be an issue.

In [36, 37], Hollnagel sets out to define a classification of observable errors, or what he calls *error phenotypes*. Rather than relying on an analysis of actions and plans in a limited domain, he begins by enumerating all possible errors involving a single action that can occur in a generic plan, defined as a totally ordered sequence of actions all of which address a single goal.

The phenotypes are classified according to the level of observation or inference needed to identify them. *0-order* phenotypes are those that can be detected based on the observation of a single action together with an expectation for what the next action will be. The more complex *1-order* phenotypes are derived from the combination of two or more 0-order phenotypes.

The identification of errors from observation of a sequence of actions requires reasoning about the temporal constraints on actions. There are two separate but related factors to consider here: the *relative* ordering of actions with respect to each other, and the *absolute* constraints on when actions must be done with respect to the real time line. The 0-order phenotypes are divided into those that can be identified on the basis of the ordinal sequence of actions, and those that are defined in terms of external temporal constraints.

Sequence-based error phenotypes

Hollnagel's error phenotypes are defined in terms of a generic plan consisting of a totally ordered sequence of actions, $[Step_1, Step_2, \ldots, Step_n]$, which is assumed to be necessary and sufficient to achieve a goal G. In the definitions that follow, the assumption is made that a single goal is being pursued at a time. It is also assumed that actions in a plan are executed one at a time, and that the results of an action are available immediately.

On the other hand, there is no assumption made about how the current goal, G, is determined – whether it is selected by the actor and inferred by the observer who would then interpret subsequent actions in terms of whether they contribute to the actor's initial goal, or whether the observer interprets all actions with reference to a goal that she herself has determined to be relevant. In either case, it is possible for the goal to change at any time as a result as information gained during the plan execution process.

The complete set of 0-order error phenotypes that are based purely on sequence are:

- Correct action: In any classification of erroneous action, it is necessary to define when an action is *not* in error. A correct action is defined as an action that is correctly placed in the currently executing plan. If $Step_i$ has just been executed, then $Step_{i+1}$ if seen next would be considered a correct action.

- Jump forward: An action that belongs further forward in the plan than the expected next action. In the observed action sequence $[\ldots, Step_i, Step_{i+2}, \ldots]$, $Step_{i+2}$ is considered a jump forward because it is being done before $Step_{i+1}$.

- Omission: This is defined as a jump forward of just one action.

- Jump backward: An action reverts to an already executed part of the plan. For example, in the observed sequence $[\ldots, Step_{i-2}, Step_{i-1}, Step_i, Step_{i-2}, \ldots]$, the second occurrence of $Step_{i-2}$ is a jump backwards because it appears earlier in the plan.

- Repetition: A jump backwards of just one action, so that the last action is repeated.

- Intrusion: the occurrence of an extraneous action in a plan. If the expected sequence is $[\ldots, Step_1, Step_2, Step_3, \ldots]$ and what is observed is $[\ldots, Step_1, X, Step_3, \ldots]$, where X is an action that does not appear anywhere in the plan for G, an intrusion has occured.

Absolute time-based error phenotypes

The absolute time constraints on actions arise from the situation and intrinsic properties of the actions themselves rather than their relationships with other actions. These properties include the urgency of the goal being addressed by an action and the duration of the action, which together can be used to specify the latest possible time the action can be started in order to have it completed in the time available to address the goal. If the goal is to change the value of a fluent variable (e.g. reducing pressure in the chest cavity), the amount of time between *onset* of the action (e.g. aspirating the chest) and initial achievement of the goal may also be relevant.

Hollnagel bases his time-based error phenotypes on the formalism for reasoning about actions and time that was developed by Allen [1]. In Allen's logic, actions are defined as events that are caused by agents and that occur over specific intervals of time. There is a small set of relationships that can exist between two time intervals X and Y:

- BEFORE(X,Y): X is completely before Y and there is a non-zero interval between them.

- EQUAL(X,Y): X and Y are the same interval.
- MEETS(X,Y): X is completely before Y but there is no interval between them.
- OVERLAPS(X,Y): X starts before Y and Y ends after X.
- DURING(X,Y): X is fully contained in Y.
- STARTS(X,Y): X and Y begin at the same time, but X ends before Y.
- FINISHES(X,Y): X and Y end at the same time, but X starts after Y.

In order to talk about actions with flexible time constraints on when they should start and finish, each action has associated with it an Earliest Starting Time (EST), a Latest Starting Time (LST), an Earliest Finishing Time (EFT), and a Latest Finishing Time (LFT). Following Allen, these time "points" are treated as extremely small intervals. These constraints, together with the temporal interval relationships above, are used to define an additional set of 0-order phenotypes in terms of absolute temporal constraints. These errors all assume that the action in question is the correct next action in the sequence, but that it is being done at the wrong time. In the following definitions, A denotes the interval during which the action occurs:

- Correctly timed action: BEFORE(EST,A) $\wedge$ DURING(LST,A) $\wedge$ DURING(EFT,A) $\wedge$ BEFORE(A,LFT). The action occurs during the correct intervals.
- Premature start of action: DURING(EST,A). The action begins before the earliest starting time.
- Delayed start of action: BEFORE(LST,A). The action begins after the latest starting time.
- Premature finishing of action: BEFORE(A,EFT). The action ends before the earliest finishing time.
- Delayed finishing of action: DURING(LFT,A). The action ends after the latest finishing time.

- Omission: BEFORE(LFT,A). The action begins after the latest finishing time, effectively rendering it useless with respect to the plan. No distinction is made between the action being done too late and not being done at all since if the action is done after the LFT it will no longer be relevant and will be interpreted as an intrusion.

The 0-order error phenotypes are simple to define and can be identified immediately in a plan since they only involve a single action.[1] On the other hand, they do not allow us to get a bigger picture of what is going wrong with the plan, since the underlying cause of the error may effect a whole sequence or sequences of actions. Looking at errors involving sequences of actions, the simple phenotypes listed above can be expanded into a larger, more complex set of 1-order phenotypes. The 1-order phenotypes have the serious disadvantage that they cannot be recognized unambiguously on the basis of a single action. Since the goal of TraumaTIQ is to respond as quickly as possible to a potential error, the 1-order phenotypes are not as useful as the 0-order. Furthermore, the added complexity of the 1-order phenotypes would make them difficult to explain briefly in the context of a real emergency. For a discussion of the 1-order phenotypes the reader is referred to [37].

Hollnagel's error definitions rely on the characterization of a plan as a totally ordered sequence of actions that are necessary and sufficient for achieving a goal. This implies that once the observer has determined the actor's goal, she then knows every action that the actor should perform to achieve that goal, and in what order they should be performed. This assumption is not justified in most realistic situations for a number of reasons.

First, this characterization of plans does not take into account the possibility of alternative ways of addressing a goal. Therefore it does not allow the classification of situations in which the actor is addressing her goal, but in a sub-optimal manner. Second, it also does not consider the interactions that may occur when multiple plans are executed concurrently. For example, a realistic system must be able to recognize which of the currently active plans an action is intended to participate in.

Finally, while Hollnagel defines plans in terms of totally ordered action sequences, in general, the actions in a plan do not have to be totally ordered. It may be up to the agent

[1] This is true provided the EST, LST, EFT and LFT are not dependent on other actions in the plan. For example, actions may be done temporarily "buy time," increasing the amount of time available for definitive action.

executing the plan to decide what to do first, and some actions may be done in parallel. The relative ordering of actions in a plan can be constrained by a number of relationships or interactions, including precondition achievement, logistical or resource constraints, and avoidance of contraindications. Violations of these constraints define a certain class of erroneous action.

5.3 Outcome-driven Error Classification

The main concern in developing a taxonomy of error phenotypes, as Hollnagel notes, is to constrain the definitions to rely only on observable findings. This will allow the definitions to be operationalized in the implementation of computer systems designed to detect and respond to erroneous actions. Thinking about plans abstractly as generic sequences of actions can help to develop a complete classification of the types of errors that *may* occur in any plan. Applying these definitions to a real domain involves thinking about which of them are relevant to the problem.

Ultimately, our goal in providing automated real-time decision support is not to identify errors when they occur, but to *prevent them from occuring in the first place*. This requires the ability to recognize and respond to potential errors as quickly as possible, while they still may be preventable. Since some of the error types defined by Hollnagel, such as *premature start of action*, cannot be identified until they are committed, we will not be concerned with these error types in TraumaTIQ.

In order to anticipate potential errors, TraumaTIQ must rely on the information available from the physician's *orders* for actions rather than waiting until the action is performed. Because of this, we must keep in mind that there will be less information available about when actions will actually be performed and what order they will be done in. The time an action is ordered only constrains the earliest possible starting time of the action. An arbitrary amount of time may pass between an order being placed and the action being performed. In addition, the actions that have been ordered so far may be performed in any order with respect to each other.

TraumaTIQ recognizes three basic discrepancies that can occur between a physician's

proposed plan and the target plan constructed by TraumAID. Of the time-based discrepancies, we are concerned only with instances of *delayed start of action*. We divide the sequence-based error phenotypes into two broad categories: *unexpected actions* and *scheduling errors*.

Omission Theoretically, an error of omission occurs when an action appears in the target plan but not in the proposed plan. In practice, we identify a *potential* error of omission when observing a *delayed start of action*, where the action does not begin until after the latest starting time. Of the remaining time-based errors, *premature start of action*, *premature finishing of action* and *true omission* (when the action does not begin until after the latest finishing time) are not useful categories because these errors are not preventable once they have been identified. The remaining time-based error type, *delayed finishing of action*, is not a significant problem in trauma management, assuming that the action is correct and is correctly executed (a requirement that TraumaTIQ cannot verify).

Unexpected Action An action that appears in the proposed plan but not in the target plan is an unexpected action. This corresponds to Hollnagel's categories of *intrusion*, *jump backwards*, and *repetition*. At this level, we are not concerned with whether the action has been done before, just that it is not correct at the present time. Unexpected actions can play a useful role in the plan, serving as an alternative way of addressing a relevant goal, in which case they are treated as *procedure choice errors*. Otherwise, they are treated as true *errors of commission*.

Scheduling Actions that are done in a different order in the proposed plan than in the target plan represent a scheduling error. 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 a judgment as to when an error of this type is actually likely to occur. To minimize intrusiveness, TraumaTIQ withholds its comments if it is *possible* that the correct scheduling is intended. If TraumAID has a constraint to do A before B, and the physician has ordered B and then ordered A, TraumaTIQ does not identify this as a scheduling error, since it is impossible to determine the actual order of execution. On the other hand, if the physician has ordered B and

not A, a scheduling error is noted. By this definition, observations of the sequence-based errors of *omission* or *jump forward* are recognized as potential violations of scheduling constraints, since a later action is being done before an earlier action in the plan.

Rather than identifying more complex permutations of erroneous action sequences, further classification of these basic errors is based on the *potential impact* of the error on the *outcome* of the plan. This is a feature that is not specified in other error classification systems I have been able to discover, but which is crucial to the ability to respond appropriately to erroneous actions. For example, an error of commission in which the action may need to be done in the near future is more tolerable than an error of commission for which it has been determined that the action should never be done. Each sub-category has a different evaluation function associated with it for calculating the error's level of significance. The relevant influences vary depending on the type of error, as I discuss in the following subsections. The entire taxonomy of errors is shown in Figure 5.2.

The taxonomy is *sound* in that it does not misclassify errors. Any discrepancy that is found by TraumaTIQ between the physician's plan and TraumAID's plan will be treated appropriately. On the other hand, the taxonomy is not *complete* because it does not include discrepancies that are not specifically of interest to TraumaTIQ. For example, as I mentioned above, TraumaTIQ does not recognize time-based errors other than delayed start of action because the other time-based error types either cannot be prevented by a critique (premature start or finish of action and true omission) or cannot be identified using TraumAID's current representation of actions (delayed finish of action). In addition, other basic error types are treated in the same class by TraumaTIQ since they are functionally equivalent from the point of view of potential outcome. Furthermore, the classification of scheduling errors is limited to the scheduling constraints that are relevant to trauma management. Clearly, other domains may have different constraints on the scheduling of actions.

It is important to recognize that this taxonomy is just one way of classifying erroneous actions. Any classification scheme must necessarily be driven by the purposes of the designer, which in this case are to determine the potential significance of errors on patient

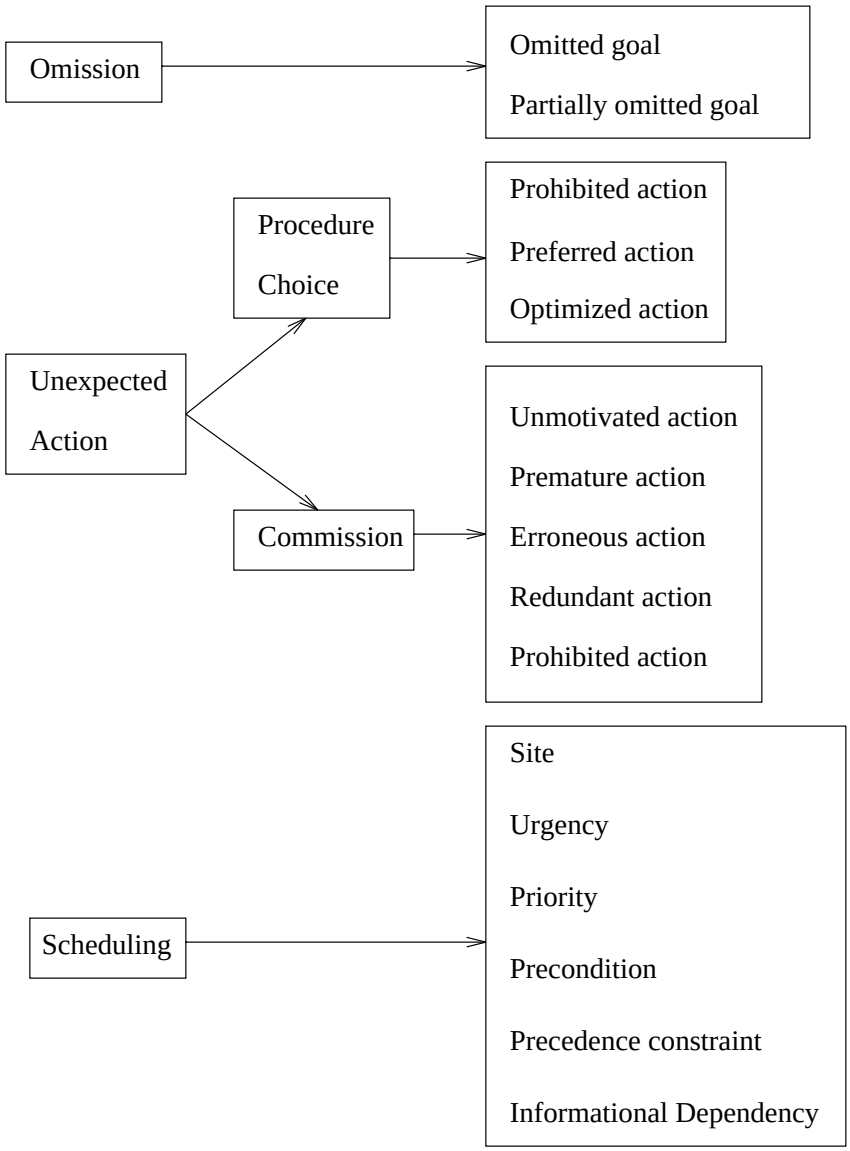

Figure 5.2: The Outcome-Driven Taxonomy of Errors

outcome. Thus I am not claiming that this classification is unique – different problems may require a different basis for classifying errors. However, classifying errors in this way would be useful in any system that is designed to tailor its responses based on the potential seriousness of what it is responding to. Any system that provides *quality assurance* would belong in this category.

5.3.1 Errors of omission

In a situation that involves pursuing multiple goals, we must consider not just what *actions* have been omitted in the plan to address a single goal, but also what relevant *goals* are not being addressed. An error of omission is identified whenever a goal that TraumAID considers relevant is not being addressed by the physician in a timely manner. Errors of omission are further classified according 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.

In either case, all the actions that are missing from the physician's plan are included in the resulting critique in connection with the omitted goal.

TraumaTIQ identifies a partial omission when a goal recommended by TraumAID is only being partially addressed using the preferred procedure. It *does not* treat as errors of omission cases in which the physician has partially pursued an unmotivated goal (errors of commission), or partially executed an incorrect procedure (procedure-choice errors). This is because of a design decision not to have TraumaTIQ assist physicians in pursuing sub-optimal or incorrect courses of action. Rather, the response to an unexpected action being ordered will be to critique the order until it is either rescinded or the action is executed.

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, TraumaTIQ identifies a latest starting time (LST) for the action as a certain period of time after TraumAID

concluded the goal to be relevant. If the LST passes without the goal being addressed, TraumaTIQ identifies an error of omission.

The amount of time between TraumAID's goal formation and the LST for addressing the goal depends on the urgency of the goal. A goal's urgency is classified as either catastrophic, unstable, or stable. These categories correspond roughly to having either 2, 20, or 200 minutes respectively in which to address the goal. TraumaTIQ sets the LST for catastrophic goals to be immediately after the goal is concluded, unstable goals two minutes later, and stable goals five minutes later. This is the amount of time after TraumAID has concluded a goal to be relevant that it will wait before commenting if the goal is not being addressed.

The omission routine is also sensitive to whether the goal underlying an action is *dependent* on any actions that are scheduled to be done before it. An action, α, is considered to be dependent on another action, α', if there is some possible outcome of α' that could cause the goal/s motivating α to become irrelevant.[2] If it is possible that an action will be removed from TraumAID's plan before it is done, it will not be mentioned as an error of omission in the critique.

5.3.2 Unexpected actions

When the physician orders an action that is not in TraumAID's plan, how it is handled in the critique depends on whether it addresses a goal that is in TraumAID's plan. If so, it is treated as a *procedure choice* error – the goal is correct, but the manner of addressing it is not. If, on the other hand, the action is not associated with a relevant goal, it is treated as an *error of commission*.

Procedure-choice errors

A procedure choice error is an unexpected action which addresses a goal that is also currently pursued in the target plan. Procedure choice errors are classified according to the reason TraumAID selected the procedure it did to address the goal. There are three reasons that TraumAID may have for choosing one alternative procedure over another:

[2] Because of the *unless* clauses in TraumAID's rules, it is possible for a relevant goal to become irrelevant as a result of new information.

1. TraumAID selected a less preferred procedure due to the presence of a scheduling constraint or contraindication for the preferred procedure.

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

3. 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 (action overloading).

When the physician and TraumAID disagree about how to address a goal, TraumaTIQ will point out that disagreement together with TraumAID's reason for having made the decision it did.

There is a slight complication here in that the third case can occur at the same time as either of the first two cases. The preferred procedure for a goal may be contraindicated at the same time that using another procedure results in a more optimal plan, or the preferred procedure may also be used to address some other relevant goal. Because the concept of action overloading requires a more complex and lengthy explanation, if either of these combinations occur, the action overloading explanation will be left out of the critique in favor of the simpler explanation.

For example, the goal of ruling out abdominal bleeding can be addressed by a peritoneal lavage or a CT scan, but the lavage cannot be done if there is extensive abdominal scarring. If the physician orders a lavage for a patient with a scarred abdomen, and there is also another reason for doing a CT scan (ruling out a renal injury, for example), the system will suggest doing a CT scan rather than a lavage to rule out abdominal bleeding on the basis of the lavage being contraindicated, without mentioning that the CT scan will also address the possibility of renal injury.

Errors of commission

An error of commission is identified when an action is observed that does not have a relevant goal associated with it. If any goal (or goals) has been selected as the reason for ordering the action[3], this routine will seek to determine *why* that goal is not currently relevant.

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

There are six subcategories of errors of commission:

1. The plan recognizer failed to infer a goal or goals underlying the action.

2. The goal associated with the action is potentially relevant, implying that the action is being done prematurely.

3. The goal has been found to be irrelevant by the failure of all of TraumAID's rules associated with that goal.

4. The goal has already been addressed.

5. The goal is relevant, but TraumAID's planner was unable to address it in the plan due to some conflicting constraint.

Since the plan recognition algorithm infers relevant goals to explain actions whenever possible, an action will not be identified as an error of commission unless there is *no* possible reason for doing it that is currently relevant. The only way the third, fourth or fifth type of commission will be identified is if the goal in question is the *only* possible reason for doing the action. Otherwise the goal is left unspecified by the plan recognizer, and the error is found to be the first type of commission.

In the case of the sixth type of error of commission, if TraumaTIQ is going to critique an action as an error of commission on the grounds that it could not be scheduled in the current plan, it is very important to be able to explain to the physician *why* it couldn't be scheduled. For this purpose, I have modified the planner to keep a record of such planning failures, so that they could be explained in the critique. This modification will be discussed further in Chapter 7.

5.3.3 Scheduling errors

The scheduling routine is concerned with enforcing temporal ordering in the plan. A scheduling error is identified when an action is ordered that is constrained to be done after another action that has not yet been ordered. This decision to consider a scheduling error only if it involves an omitted action (in the sequence sense of omission) means that the

critique will often include an error of omission along with a scheduling error. In Chapter 6 I will discuss the possibility of combining such related comments.

Scheduling errors are classified according to the *reason* for the scheduling constraint. The possible reasons are as follows:

1. Site: Since TraumAID plans for the patient to be moved through the necessary sites in a fixed order – Emergency Center, Radiology Suite, Operating Suite, Trauma Unit – actions that must be done in an earlier site are constrained to be performed before actions that must be done in a later site.

2. Urgency: Each action has a level of urgency associated with it that is inherited from the goals it addresses. An action α cannot precede another action α' in the plan if α takes more time than the urgency level of α' indicates is available.

3. Priority: If there are no differences in urgency, standards of trauma practice recommend addressing problems in order of the subsystem of the body they affect: airway before breathing before circulation, etc.

4. Precondition: Certain actions have preconditions that must be satisfied before they can be done. In TraumAID's plans, these preconditions are enforced implicitly by the ordering of actions within procedures. For the purposes of critiquing, I have added explicit preconditions to TraumAID's action definitions, as described in Chapter 7.

5. Precedence constraints: Some combinations of actions must always be done in a predefined order because the performance of one could affect the outcome of the other.

6. Informational Dependencies: Since the plans produced by TraumAID are conditional plans, in some cases the relevance of a goal that is addressed later in the plan may be dependent on the results of information to be acquired earlier in the plan. In that case, we want to be sure that the actions are done in the appropriate order, since the later actions may turn out to be altogether unnecessary.

When a scheduling error is commented on in the critique, it will include the reason that TraumAID has scheduled one action before the other.

Note that the *precondition* category is unique in that it is the only time that TraumaTIQ will recommend doing an action that is not in TraumAID's plan. This will be the case if an unexpected action, α, with an unsatisfied precondition, β, is ordered. In addition to an *error of commission*, TraumaTIQ will identify a *scheduling error* regarding the precondition. The resulting critique will say both "Do not do α," and "Before doing α do β."[4] This contradicts the general philosophy that TraumaTIQ should not assist physicians in executing incorrect plans, because in this case the consequences of doing α without doing β are great enough that the physician should be made aware of them.

5.4 Determining the significance of errors

To provide effective decision support in a time-critical, task centered activity such as trauma management, the plan evaluator must be designed with the cognitive demands on the physician in mind. For this reason, TraumaTIQ limits its critique to those items that may have a significant negative impact on the outcome of the case. Unlike other critiquing systems [54, 70], TraumaTIQ does not comment on correct decisions. While it may be desirable in some situations to encourage the user with positive feedback, in an urgent situation the need to reserve the physician's attention for the primary task outweighs such psychological benefits. Therefore, in TraumaTIQ the absence of a critique will be taken as acceptance of the proposed plan.

Beyond not commenting on correct actions, the approach I have taken also refrains from producing a comment when the physician's proposed plan diverges only in minor ways from the plan recommended by TraumAID. In this section I will discuss how TraumaTIQ determines when a divergence is significant enough to warrant a critique.

The first two stages of plan evaluation identify and classify places 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 blood or other fluids in the abdominal cavity). This would seem to be a significant

[4] This is another example of critiques that should ultimately be combined in the output (e.g. "α is not recommended, but if you decide to do α anyway, be sure to do β first.")

error since it has costs in terms of both time and invasiveness. On the other hand, an unmotivated administration of intravenous fluids would not be considered as important to correct since the costs involved are not high.

While TraumAID's knowledge base has been carefully designed to reflect nationally accepted practice guidelines, in some cases it may be possible to stray from these guidelines without incurring an unacceptable amount of additional cost or risk to the patient. In such cases, it is more important to allow the physician to attend to the task of patient management than it is to correct her minor deviations from protocol. In addition, some physicians may have more experience performing one procedure than another, and following the practice guidelines may not be the optimal approach in such cases.

There are two reasons for incorporating this knowledge into the system:

1. It will reduce the number of comments in the critique that reflect insignificant differences between the preferences coded in TraumAID's knowledge base and the preferences of the physician using the 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.

2. Errors that are important enough to appear in the critique will still vary as to their potential impact on the patient. It is important for the system to be able to identify those errors of a particularly serious nature and emphasize them in the critique.

The third stage of plan evaluation evaluates the error instances classified during the second stage according to knowledge about the clinical significance of various types of error. For the purposes of critiquing, each error is classified as either:

1. Tolerable, probably harmless.

2. Non-critical, but potentially harmful.

3. Critical, potentially fatal.

Which of these classes a particular error belongs to determines how it will be handled in the critique. Errors in the first class are not mentioned at all, errors in the second

class appear as simple statements or reminders, while errors in the third class appear as warnings.[5]

5.4.1 Representing disutilities for errors

In order to evaluate the significance of individual errors, it is necessary to determine *how much worse off* the patient will be as a result of those errors. To do this we can use the decision-theoretic concept of *disutility* [76]. Decision theory is usually concerned with finding the course of action that will *maximize* expected utility. In this case, we are interested in the *difference* in expected utility between the physician's proposed plan and TraumAID's plan. If that difference is sufficiently high it will motivate producing a critique.

In the absence of a "gold standard" for evaluating trauma management plans, I took advantage of a set of disutilities, or negative utilities, associated with actions and outcomes that was generated during the development of the TraumAID system. The procedure for determining these disutilities was as follows:

The subjects were four surgeons who were knowledgeable about the procedures and outcomes relevant to the management of trauma. These physicians were asked to make judgments from the point of view of a patient undergoing trauma care. There were two potential problems with this procedure: the physicians' perception of outcomes is probably somewhat different from the average trauma patient simply on the basis of greater experience, and their judgments may also be biased in that they do not represent a random sample of society. However, the advantage of their greater knowledge and experience was seen to outweigh potential biases for the purposes of this task.

The subjects were presented with a list of trauma management procedures and a list of adverse outcomes (failures to address diagnosed problems). They were asked to rank each item on a scale of 0-100 where 0 is best and 100 is worst, on the basis of how they would feel about having to undergo that procedure or experience that outcome. These judgments incorporated assessments of pain and discomfort, recovery time, and prognostic implications. They did not, however, include consideration of financial cost, since they

[5]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.

were intended to be used primarily to maximize the patient's physical well-being.

After the initial ranking, the disutilities were refined using an iterative series of standard gamble comparisons. In a standard gamble a subject is asked to choose between a definite intermediate outcome or a probabilistic outcome which is either better or worse. For example, a subject might be asked to choose whether they would rather have a 100% chance of getting $50 or a 50% chance of getting either $100 or nothing.

To determine relative disutilities between all procedures and outcomes in the trauma domain, a sequence of standard gamble comparisons was necessary. To begin, the worst possible outcome was anchored at 100, and the best outcome was anchored at 0. Starting with the worst outcome, an item was chosen that had initially been assigned a disutility of half of that outcome. The subject was then asked whether they would rather experience the one with the lower disutility or have a 50% chance of experiencing the one with the higher disutility. If a preference was indicated, the probability of experiencing the higher disutility was adjusted until the subject did not have a preference. The relative disutility of the lower item was then adjusted to reflect this new value. For example, if the subject did not have a preference between definitely having a tube thoracostomy and having a 10% chance of having an ER thoracotomy, then the disutility of the tube thoracostomy would be adjusted to be equal to 10% of the disutility of the ER thoracotomy.

This process was then repeated with the newly adjusted item, and another item that had initially been assigned half of that adjusted disutility. In this way, procedures and outcomes were compared until a globally stable assignment of disutilities was reached. The result of this procedure was a "cost" associated with each action in TraumAID's knowledge base, and a "penalty" for failure to address each of the goals.

Action costs

The disutilities associated with procedures were assigned as "costs" to the TraumAID actions involved in those procedures. In TraumAID 2.0, these costs are used by the planner to choose between two alternative plans that address the same set of goals: given such a choice, the plan with the lower total costs is selected.

In TraumaTIQ's plan evaluation, action costs are used to determine the significance of errors of commission and procedure choice. In the former, the higher the cost of the

action being done unnecessarily, the more significant the error. In the latter, the *difference* between the costs of the alternative procedures is the relevant metric.

Penalties for omission

These disutilities relate to the failures to address problems. In contrast to the costs for undergoing procedures, these values were not used by TraumAID 2.0's planner. However, they are quite valuable for TraumaTIQ's plan evaluator to assess the significance of errors of omission. While the disutilities for undergoing procedures were translated into *costs* of *actions* in TraumAID, the disutilities for omitted treatments correspond more closely to the *goal* level of TraumAID's representation: they indicate the penalty for failing to address a therapeutic goal, regardless of what actions may be involved in addressing that goal.

5.4.2 Approximating disutilities of errors

The concept of expected disutility provides a convenient way to evaluate the significance of different types of error on a single scale. Each error is assigned a value between 1 and 100 which can then be used to decide whether the error will be considered critical, non-critical, or tolerable.

The expected disutility of a course of action is a function of both the disutility of the various outcomes and their probabilities. This can be calculated using a decision tree representing both choice (decision) nodes and chance (probabilistic) nodes. Since TraumAID does not actually calculate a numerical probability for the diagnoses it considers or for the expected outcomes of actions, it was necessary to make some assumptions about these probabilities to approximate the difference in expected disutility between TraumAID's and the physician's plans:[6]

- When TraumAID has a therapeutic goal, assume that the probability of the diagnosis is 100%. As a result, the expected disutility of failing to address a therapeutic goal is equal to the difference between the penalty for failing to address the goal and the cost of addressing it.

[6]These approximations were determined in consultation with Dr. John Clarke, an experienced trauma surgeon and decision theorist.

This also leads to calculating the expected disutility of a *redundant* diagnostic action as being equal to the cost of the action, since it definitely will not appear in TraumAID's plan in the future.

- When TraumAID has a diagnostic goal, assume that the probability of the diagnosis is 50%. This means that the expected disutility of failing to address a diagnostic goal is equal to the difference between half the penalty for failing to treat the injury and the cost of doing the diagnosis.

  This assumption also leads to calculating the expected disutility of a *premature action error* as half the cost of the action, since we have assumed a 50% probability that the action will eventually be included in TraumAID's plan.

- When TraumAID has concluded a diagnosis to be false, assume that the probability of that diagnosis is 0%. This means that the expected disutility of an error of commission of either a diagnostic or therapeutic action is equal to the cost of the action, since there is no chance that the action will be included in TraumAID's plan.

- When TraumAID has not concluded a diagnostic goal to be either relevant or irrelevant, assume that the probability of the diagnosis is 10%. This means that the expected disutility of an unmotivated error of commission is 90% of the cost of the action since there is a small chance that the action will appear in TraumAID's plan.

- Assume that all procedures for addressing a goal are equally effective. This means that the expected disutility of a procedure choice error is equal to the difference in cost between the plans containing the alternative procedures.

These assumptions allow us to calculate expected disutilities for many of the error types enumerated in the previous section. In addition, we need to be able to estimate the disutility when a goal has been *partially addressed*, and we need to know the disutility of violating the constraints that lead to action prohibitions and the scheduling constraints.

When a goal has been partially addressed by the physician's orders, the disutility should be proportional to the amount of the goal that has been addressed. Unfortunately, TraumAID's procedure definitions do not include information about how important each action is to the procedure as a whole. To approximate this value, I make the assumption

that the higher the proportion of the procedure cost contributed by an individual action, the more central a role that action has in the procedure. The disutility for partially failing to address a goal is thus calculated as

$$(\mathcal{P}_G * \frac{\mathcal{C}_A}{\mathcal{C}_G}) - \mathcal{C}_A$$

Where $\mathcal{P}_G$ is the penalty for not addressing the goal if it is relevant, $\mathcal{C}_A$ is the cost of the actions that have been omitted, and $\mathcal{C}_G$ is the total cost of addressing the goal.

The disutility of doing a prohibited action depends on the *reason* it is prohibited. This applies both to errors of commission involving prohibited actions and to procedure-choice errors where the procedure involving the prohibited action has been replaced by an alternative procedure. Actions can be prohibited by TraumAID due to either hard constraints (contraindications or lack of resources) or soft constraints (scheduling conflicts or site constraints). If an action is in violation of a hard constraint the potential disutility is very high, while violating a soft constraint does not have such serious consequences.

For scheduling errors, the disutilities depend on the type of scheduling constraint concerned: If the scheduling constraint is due to urgency, the disutility depends on the urgency of the more urgent action. If due to medical priority, the disutility is always low enough to prompt a comment but not a warning.

If the ordering is due to the fact that the second action has to be done in a later site, the disutility depends on whether the later site is the operating room, the X-ray room, or some other site. If it is the operating room, the error will be ignored because almost anything can be done in the operating room. On the other hand, if the second site is the X-ray room, the scheduling error will result in a warning, since patients often spend a long time in the X-ray room and very few actions are possible there.

If the ordering is due to precedence constraints or precondition constraints, a comment is always produced. Finally, if the ordering is due to informational dependencies, the disutility is taken to be half the cost of the action that is erroneously being done first, since it may not actually have to be done at all.

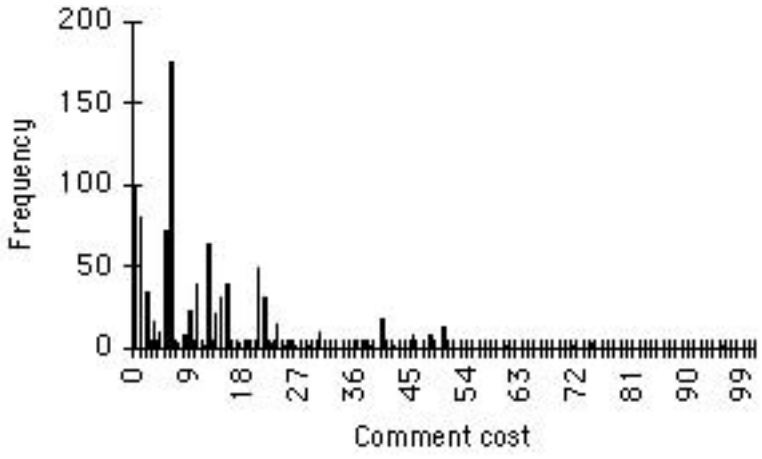

Figure 5.3: Distribution of comment disutilities

5.4.3 Thresholds for error magnitude

In order to classify the significance of errors, two disutility thresholds are needed: one for critical errors and one for non-critical errors. An error with a disutility that is above the first threshold will result in a warning, and an error with a disutility that is between the two thresholds will result in a comment. Errors with disutility values below the non-critical threshold are considered tolerable and are not mentioned in the critique.

One of the advantages of this method for evaluating errors is that it is very easy to adjust the pickyness of the system simply by adjusting the two thresholds. Ideally, the thresholds should be set in such a way as to make the system comment on every significant item during a case, while minimizing the amount of unimportant and/or unnecessary comments. For now, the threshold values have been tentatively determined by consultation with Dr. Clarke. The current thresholds are set so that a disutility between 1 and 6 will result in a comment, and a disutility of 7 or higher will result in a warning.

Figure 5.3 shows the distribution of comment disutilities for all the comments produced by TraumaTIQ on the database of 97 management plans. The mean disutility is 11.28, with a standard distribution of 12.61. The median disutility is 6. With the current thresholds for comment and warning, this means that approximately equal numbers of errors will result in comments and warnings, and very few will be ignored. The thresholds will probably need to be adjusted based on experience with physicians using the system in real cases.

5.5 Output of plan evaluation

The output of the plan evaluator is a set of comment specifications each of which is a triple, $\langle T, M, A \rangle$, where T is the comment type, M is the magnitude – either INFORM or WARN, tolerable errors are filtered from the output – and A is a list of arguments specifying what the comment is *about*.

For example, the comment

⟨proc-choice3, INFORM,
{local-wound-exploration, peritoneal-lavage, RO-abdominal-wall-injury}⟩

means that a procedure choice error was identified in which the physician is doing a peritoneal lavage to address the goal of ruling out an abdominal wall injury while the system would have chosen a local wound exploration to address that procedure. The magnitude of INFORM indicates that this error has been classified as a non-critical error.

These comment specifications become the input for the language generation phase, which is described in the next chapter.

Chapter 6

Critique Generation

The output of plan evaluation 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. In keeping with the approach seen in the language generation literature [62], 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 thesis, I have been concerned primarily with the issues associated with strategic generation – determining what to say, how to represent concepts for the purposes of language generation, and how to organize the output. I will be leaving the problem of tactical generation to others (see [68] for an approach to tactical generation, specifically the generation of contextually appropriate intonation contours, applied to the language of trauma domain), although I will consider some of the issues involved in presenting spoken versus written critiques.

6.1 Towards Strategic Generation

6.1.1 Determining critique content

In TraumaTIQ, the contents of the critique are derived from the output of the plan evaluation routines (see Chapter 8). Corresponding to the error types 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. The illocutionary force of a comment will influence the phrasing and, in the case of spoken critiques, the intonation of the output. Note that I do not adopt the strategy taken by Rankin [70] of commenting on every action proposed by the physician. While this strategy has the advantage of convincing the physician that the system has considered all of her proposals, it is probably not appropriate or necessary in a crisis management situation to confirm each undisputed action.

6.1.2 Explanations

In addition to informing the physician of potential errors in her plan, justifications must be included in support of important points. The importance of explanation for enhancing the acceptance of expert systems is well known [86]. Techniques for generating explanations based on the knowledge and reasoning process have been described in [57, 60, 61, 85]. It is also understood that explanations should be tailored to the user's current goals [9, 88].

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 the 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 an X-ray of the abdomen. This is because of the urgency of treating the hemothorax."

This comment presupposes that the physician knows that a massive hemothorax must be attended to urgently, but suggests that she 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 knowledge base of TraumAID 2.0. Future extensions of the system might include more explicit medical knowledge, so that the system could present more detailed explanations at the level of basic biomedical reasoning if desired.

In addition to the explanations included with critiques, TraumAID also has the capacity to provide interactive explanations of its reasoning at the request of the user. At any point, TraumAID can answer queries regarding what rules lead to a particular conclusion or goal, or what goals it is addressing by including a particular action in the plan.

6.1.3 Repetition or duration of critiques

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 her current course of action in spite of a critique, it is probably necessary to repeat the comment since she may not have heard it or paid 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.

In the case of visual text presentation of the critique, the question is how long a comment should remain on the screen after it is first displayed if it remains relevant. Currently, TraumaTIQ leaves comments on the screen as long as they remain relevant. However, a policy of removing comments after a short period and removing warnings after a somewhat longer period is probably motivated under the assumption that the more clinically significant the information contained in a comment, the more important it is to make sure the physician is aware of it. The appropriate length of time to persevere with a comment will be resolved through on-site experimentation with the system.

6.1.4 Structure of the critique output

To emphasize the more significant comments, TraumaTIQ displays all warnings before other comments. To further organize the comments making up the critique, TraumaTIQ

has a "topic slot" for each comment type it produces. The topic of a comment is the concept filling the topic slot. The topics are determined as follows:

- For errors of omission, the topic is the goal that is not being addressed.
- For errors of commission, the topic is the action that has been ordered.
- For procedure choice errors, the topic is the goal being addressed by both procedures.
- For scheduling errors, the topic is the action that is supposed to be done first.

Comments are sorted by topic so that all comments about the same concept are grouped together and presented 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 (cf. [55, 70]).

There are, however, situations in which multiple comments may be combined to improve the coherence of the overall critique. For example, the following two comments may be produced in a case with a possible fractured vertebra and a possible abdominal injury, in which the physician has ordered a peritoneal lavage but not a lateral chest x-ray.

> "Consider doing a lateral chest X-ray to rule out a fractured vertebra."
>
> and
>
> "Do a lateral chest X-ray before doing the peritoneal lavage because the latter may affect the results of the former."

These two comments are related in that they are both about doing a lateral chest X-ray, and can be combined into a more compact (but longer) statement:

> "Consider doing a lateral chest X-ray to rule out a fractured vertebra. Do it before doing the peritoneal lavage because the latter may affect the results of the former."

This example also points out the discourse-related issue of selecting appropriate referring expressions, including determiners and pronouns. The second sentence of the first

example should really say "do *the* lateral chest X-ray," not "do *a* lateral chest X-ray," since the procedure was referred to in the preceding sentence. In the second example, the pronoun "it" is used to refer to the aforementioned X-ray.

Another example of a situation in which multiple comments can be combined is in the case of multiple errors of omission. Errors of omission in TraumaTIQ refer to goals that are being omitted, so if the same action is suggested for addressing more than one goal, two separate comments will be produced. In such a situation, the number of comments could be reduced by combining all comments regarding errors of omission that suggest doing the same action or actions into one comment. For example:

> "Consider getting a chest X-ray to rule out a hemothorax, rule out a pneumothorax, and evaluate the airway."

6.2 Towards Tactical Generation

The tactical generation of natural language from a semantic representation of propositional content is an important area of research in its own right, and one that I have not set out to solve in this dissertation. This section describes the template filling algorithm that I have used in TraumaTIQ to generate natural sounding sentences. I will also include 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.

6.2.1 Templates

The first component of sentence generation in TraumaTIQ is a set of templates corresponding to the comment types identified by the plan evaluation module. For each comment type, there are two templates, one for warnings and one for statements. The templates are as follows:

- Errors of Omission:
  - omission of some actions in a procedure:
    - INFORM: "Consider _ as part of _."

    * WARN: "Caution: _ immediately as part of _."
  - failure to address a goal:
    * INFORM: "Consider _ now to _."
    * WARN: "Caution: _ immediately to _."
  - omission of bedside question(s):
    * INFORM: "Consider checking for _ to assess the possibility of _."
    * WARN: "Caution: check for _ to assess the possibility of _."

- Errors of Commission:
  - unexplained action:
    * INFORM: "_ seems unmotivated. Please reconsider this action."
    * WARN: "Caution: _ is not justified based on the information currently available."
  - premature pursuit of goal:
    * INFORM: "_ seems premature at this point. There is not yet enough information to justify _."
    * WARN: "Caution: _ is premature. There is not yet enough information to support _"
  - erroneous pursuit of goal:
    * INFORM: "_ seems unmotivated because _ has been proven to be unnecessary."
    * WARN: "Caution: _ is not justified because _ has been proven to be unnecessary."
  - redundant pursuit of goal:
    * INFORM: "_ seems unmotivated because _ was already addressed."
    * WARN: "Caution: _ is not justified because _ was already addressed."
  - unmotivated pursuit of goal:

    * INFORM: "_ seems unmotivated. There is not enough information to conclude the relevance of _."
    * WARN: "Caution: _ is not justified. There is not enough information to support _."
  - unperformable action:
    * INFORM: "Do not _ now because _."
    * WARN: "Caution: do not _ now because _."

- Procedure Choice Errors:
  - alternative to unperformable action:
    * INFORM: "Please consider _, rather than _, to _. The latter cannot be done because _."
    * WARN: "_, rather than _, to _. The latter cannot be done because _."
  - default preference:
    * INFORM: "_ is preferred over _ for _."
    * WARN: "_ is highly preferred over _ for _"
  - optimization:
    * INFORM: "Please consider _ rather than _ to _. The former can also be used to _."
    * WARN: "_ rather than _ to _. The former can also be used to _."

- Scheduling Errors:
  - urgency:
    * INFORM: "Please _ before _ because it is more urgent."
    * WARN: "Caution: _ before _ because it is very urgent."
  - medical priority:
    * INFORM: "Please _ before _ because it has a higher priority."
    * WARN: "Caution: _ before _ because it has a very high priority."

- site constraint:
  - INFORM: "Please _ before going to _ to _."
  - WARN: "Caution: _ before going to _ to _."
- precondition:
  - INFORM: "Please remember to _ before _."
  - WARN: "Do not forget to _ before _."
- precedence constraint:
  - INFORM: "Please _ before _ because the latter may affect the results of the former."
  - WARN: "_ before _ because the latter may affect the results of the former."
- informational dependency:
  - INFORM: "Do not _ until you have _. The outcome of the latter may affect the need to do the former."
  - WARN: "Caution: do not _ until you have _. The outcome of the latter may affect the need for the former."

6.2.2 Filling the slots in the templates

The slot fillers for the templates are constructed from TraumAID's action, procedure, and goal concepts. For each of these concepts in TraumAID's knowledge base, I have associated a phrasal translation. For example, the action Close_Chest_Wound is translated as "*close $a chest wound." The asterisk before the word "close" indicates that it is a verb that needs to be conjugated. The string "$a" indicates a determiner that will be realized either as "the" or "a" (or "an") depending on the status of the concept containing it in TraumAID's current representation of the case (and the next word).

There are three different kinds of noun phrases that appear in the translations of TraumAID's goal, procedure, and action concepts: (1) anatomical parts (eg. *the heart*), (2) action names (eg. *a urinalysis*), and (3) injuries (eg. *a/the lacerated diaphragm*). As I have indicated in these examples, anatomical parts will always get a definite article since their presence is assumed to be common knowledge, while action names always get an

indefinite article because they are generally being introduced into the discourse by the critique.[1]

References to injuries are either definite or indefinite depending on whether the comment assumes that the presence of the injury is common knowledge. For that reason, injuries that are the object of diagnostic goals are always indefinite. Injuries that are the object of therapeutic goals that are not supported by TraumAID are referred to indefinitely (eg. "Covering *a* chest wound is unjustified at this time. There is not enough evidence to support treating *an* open sucking chest wound"). On the other hand, therapeutic goals that are supported by TraumAID result in a definite reference (eg. "Consider covering *the* chest wound now as part of treating *the* open sucking chest wound."), implying that the system believes that the physician is aware of the diagnosis, but has just forgotten to act on it.[2]

Each slot in a template has a label indicating how the main verb or verbs in the phrase it contains should be conjugated. In the first template, for example, the first slot filler is to be realized as a gerundive phrase (eg. "Consider [*getting* a chest X-ray]..."), while the second is untensed (eg. "...to [*rule out* a hemothorax.]")

A slot may be filled with a single concept, or with a list of concept, in which case the list will be marked as either conjunctive or disjunctive and realized as a list of translated phrases separated by either "and" or "or." For example, "Consider checking for medication allergies, giving antibiotics, and doing a laparotomy now to treat the lacerated diaphragm."

6.2.3 Generating spoken critiques

Currently, TraumaTIQ displays its critiques as single-sentence comments produced by inserting specific action and goal names into stored templates.

In [67], Prevost and Steedman describe an approach to tactical generation using a Combinatory Categorial Grammar (CCG) formalism. 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

[1] If a concept is mentioned more than once in a set of comments, it should really get a different referring expression after the first time, but this is a discourse issue that I have not dealt with here.

[2] The issue of politeness in phrasing the critiques is an important and delicate matter that I will discuss briefly later in this chapter.

proposition is about) and *rheme* (what new information the proposition has to say about the theme) are marked, a surface string is generated 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 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.

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, could then be used to generate sentences conveying any number of propositions that we may decide should be included in the critique.

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[3] 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 with prosodic stress in a spoken critique.

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

Chapter 7

TraumaTIQ: a Real-Time Critiquing Interface for Trauma Care

The ideas presented in the preceding three chapters have been implemented in TraumaTIQ, the critiquing interface for TraumAID. TraumaTIQ is implemented in Common Lisp and runs in conjunction with TraumAID on both Unix/X-Windows and Macintosh platforms.

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, TraumaTIQ produces a series of critiques as patient management progresses. The critiques are 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

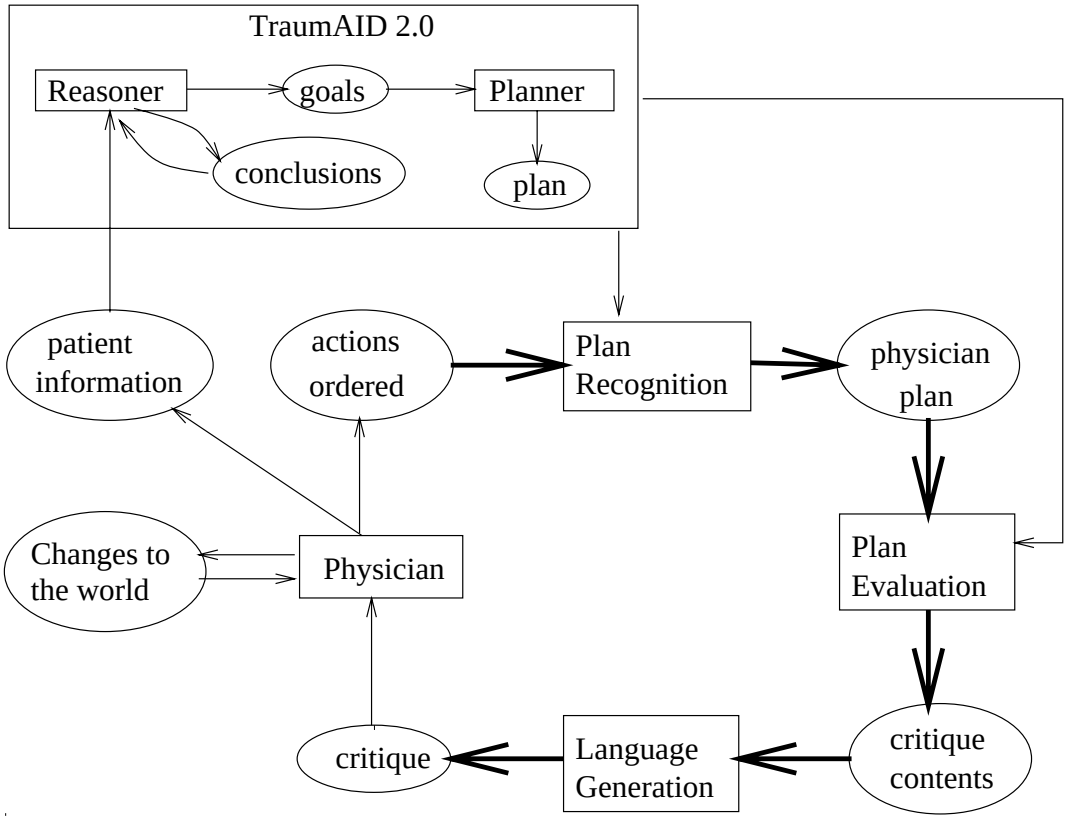

Figure 7.1: The TraumaTIQ module

physician. Critiques are generated based on the complete set of 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 an object of the critique, whether or not it was appropriate.

The architecture of TraumaTIQ is shown in Figure 7.1, which was seen earlier at the end of Chapter 2. TraumaTIQ's plan recognizer is an implementation of the algorithm presented in Chapter 4. It uses knowledge about the situational context and about the plan so far to control the search for explanations of new orders.

The plan evaluation module first compares the explanatory plan constructed by the plan recognizer with the target plan constructed by TraumAID and identifies errors in the plan according to the classification described in Chapter 5. These errors are then evaluated to determine their potential significance to the outcome of the plan. Only errors that have

the potential to affect the outcome are reported in the critique.

The critique generator converts the output of the plan evaluator into English sentences, using a simple template filling procedure which generates natural sounding sentences by adjusting the slot fillers depending on the template they are filling. Finally, the sentences are sorted by significance and topic, and displayed in a window on the monitor. In the Macintosh implementation, the critique sentences are also processed by the Macintosh Speech Manager and output as synthesized speech.

The next section presents a discussion of the changes to TraumAID's knowledge base and planner that were necessary for the implementation of TraumaTIQ. Following this discussion the next section goes through an example of a actual trauma case, and how TraumaTIQ would have responded at each point had it been operating during the case.

7.1 Changes to TraumAID

Because we already had a planning system that produced validated management plans and an extensive knowledge base representing conclusions, goals, and actions in the domain, it seemed natural to use the knowledge base and representation of plans from TraumAID verbatim for the plan recognizer as well. Not only could TraumAID tell us what goals were possible explanations for the actions we observed, but it could also tell us what goals were more *likely* to be pursued in the current context, under the assumption that physicians are likely to pursue relevant goals. The main disadvantage of this approach is that a knowledge base that is sufficient for generation of valid plans is not necessarily complete in terms of the plans that people actually carry out. We therefore have had to incorporate additional knowledge about goals and plans that, while they would not be produced by TraumAID, are likely to be seen in actual patient management.

Implementing the system necessitated several changes and additions to the core TraumAID system. These changes occurred in two places: the knowledge base, and the planner. In the next two sections I will describe these additions.

7.1.1 Knowledge representation

The knowledge base used by TraumAID was designed to generate complete and efficient plans. Since the range of possible plans can be much less constrained in the plan recognition task than in plan generation, it is more important for plan recognition that knowledge of plans and actions be represented in an explicit, declarative form [93].

Several types of knowledge that are necessary for plan recognition and evaluation were not represented explicitly in the knowledge base of TraumAID 2.0, as it was not designed to support those functions. Other types knowledge were not included at all. The following features were added to TraumAID's knowledge base to support the implementation of TraumaTIQ.

7.1.2 Conceptual links between goals

Since TraumAID's planner generates a new plan each time new information is entered about the case, it has no reason to maintain an explicit representation of the relationships between goals over time. Consequently, no explicit connection was made between diagnostic actions and their related therapeutic actions in TraumAID's knowledge base. For example, there is no explicit link between the diagnostic goal RO-HEMOTHORAX ("rule out hemothorax") and the therapeutic goal RX-HEMOTHORAX ("treat hemothorax").

Adequate critiquing, on the other hand, requires that such relationships between goals be accessible in order to understand physicians' behavior. The fact that the goal of ruling out a hemothorax is currently being pursued makes the goal of treating one more likely to be in the physician's focus of attention.

Therefore, for plan recognition links between goals were added to TraumAID's knowledge base, so that each goal is connected to all of the goals that may become relevant as a result of addressing it. This allows the plan recognizer to understand which goals are potentially relevant as a result of the current goal set.

In addition, TraumAID's reasoner has a facility for *suspecting* concepts, in order to drive the acquisition of relevant information. When a concept that was suspected becomes true, it triggers the conclusion of new goals. For plan recognition, links were added between concepts that may be suspected and the goals that would be triggered were they to be

true. When a goal's relevant concepts are suspected, the goal can be considered potentially relevant by the plan recognizer.

There is actually a related form of information in TraumAID 2.0's knowledge base called the *goal hierarchy*, which was implemented to allow the inhibition of goals when they are superseded by other goals or concepts. The goal hierarchy indicates when the knowledge of one goal or concept should cause another goal or concept to be concluded false. Most often this results in inhibiting diagnostic goals when the related therapeutic goal is concluded, or when the information sought by the diagnostic goal is already known.

Unfortunately, since its implementation was guided by the restricted needs of the planner, the goal hierarchy is not complete in that it does not represent *all* such relationships between concepts. Another drawback of the goal hierarchy is that it is compiled into TraumAID's rules when the knowledge base is loaded, and so is not available during run time. In fact, it may be the case in the future that the declarative information added to the concept definitions for plan recognition may be useful for further development of TraumAID's planner.

7.1.3 Abstract Goals

When TraumaTIQ's plan recognizer was tested off-line on actual management plans, most of the actions that were not explained were actions with broad-ranging effects: either diagnostic tests with several possible outcomes, such as X-rays, or therapeutic procedures that could be done to address several different problems, such as giving antibiotics. It is frequently the case that such procedures are done for routine screening purposes rather than to address a specific goal.[1] In such cases, a single more abstract goal that subsumed these more specific goals could eliminate any need to choose among them and provide a simpler explanation.

While the number of levels of abstraction in the knowledge base is sufficient for planning, it is not sufficiently stratified for the plan recognizer to be able to draw general conclusions when it is not able to draw more specific ones. For example, a chest X-ray may be done to rule out a bullet in the chest, rule out a simple pneumothorax, rule out a

[1] Fortunately, it is also the case that most of these actions do not incur a large cost to the patient.

simple hemothorax, rule out a lacerated diaphragm, rule out an intra abdominal gastro-intestinal tract injury, or survey the airway. Each of these goals is triggered by a different combination of findings.

In plan recognition, the aim is to identify the most specific goal or goals underlying an observed action. If no specific explanation can be found, a more general explanation may be necessary. For this reason, summarizing all of the goals motivating an action in terms of an abstract goal is desirable. In the case of the chest X-ray, all of the goals listed above can be summarized as the goal "rule out thoracic injury."

7.1.4 Additional goals

While TraumAID has all the necessary goals in its knowledge base to produce good plans, its knowledge base does not indicate all the goals that people might actually pursue while managing a trauma patient, or all the actions they might order. To recognize and appeal to such possibilities in its critiques, it is necessary to add knowledge about goals not in the original knowledge base.

For example, the action of giving analgesics (pain-killers) is done in TraumAID only as part of the procedure for treating a fractured sternum. While physicians often give analgesics in other situations, TraumAID's rules have been designed to avoid giving them in most situations because they can mask reactions that could provide useful diagnostic information. In spite of this, the plan recognition system still has to be aware that analgesics may be given for other reasons than treating a fractured sternum, which can be summarized as the goal of "treating pain."

There are other ways that the physician may act that are outside TraumAID's knowledge base. Some actions, like giving analgesics to relieve pain, are superseded by more critical goals and so are prohibited by TraumAID. Still other actions may be "optional" rather than strictly prohibited, so that while they may not appear in TraumAID's knowledge base, their appearance is not necessarily detrimental to the overall plan. Other actions may relate to injuries (or other features of the patient) that are not covered by TraumAID. In all of these cases, it is necessary to include additional goals for plan recognition.

Goal penalties

Another item missing from the knowledge base were the disutilities for omission of goals that were discussed in Chapter 5. These were added to the knowledge base for each of TraumAID's goals. Some goals were assigned a disutility directly by the expert panel. Other goals (primarily the diagnostic ones) had not been assigned a disutility since it is impossible to quantify the disutility of not pursuing a diagnostic goal without knowing the probability that the diagnosis is true. As a rough estimate, I used the assumption that if a diagnostic test is called for, the diagnosis will be present 50% of the time, and so I gave diagnostic goals half of the disutility of the related therapeutic goal. For example, if the disutility of not treating a tension pneumothorax is 100 then the disutility of not pursuing a diagnosis of tension pneumothorax, given the relevant findings of shock, decreased breath sounds, and distended neck veins is taken to be 50.

7.1.5 Scheduling of procedures

TraumaTIQ is intended to critique errors in the scheduling of actions. At any time during the management of a case, TraumAID's plan is a partial ordering, in which an action's position in the ordering may reflect its urgency, medical priority, logistics, possible interactions between actions, as well as the *a priori* ordering of actions within a procedure, as specified in the procedure's definition. Plans constructed by TraumAID's planner always conform to this ordering.

However, while the ordering of actions within a procedure is sometimes meaningful, it is sometimes arbitrary. Some actions must be done before others in order to set up the necessary conditions or insure the absence of contraindications, such as checking for allergies before administering a drug. Other actions have no such temporal relationship. For example, the procedure for treating a compound fracture of the sternum (breast-bone), calls for checking for allergies to medication and then administering both antibiotics and analgesics. For obvious reasons, the allergy check must be done before giving either drug, but the drugs can be given in any order with respect to each other.

TraumaTIQ should be able to critique a plan if an important scheduling constraint is being violated, but should not comment when the ordering is arbitrary. For this reason,

meaningful scheduling constraints have been explicitly noted in the procedure definitions, not for TraumAID itself, but as the basis for critiques regarding scheduling errors. Future work on TraumAID's planner may also use this information to generate plans more flexibly based on the presence of constraints.

7.1.6 Changes to the planner

As discussed in Chapter 6, it has often been pointed out that for a planning system to be understood and accepted by its users, it must have some ability to explain its reasoning. TraumAID 2.0 was designed for the purpose of generating quality management plans, not explaining those plans to people. Therefore, issues such as explanation and comparison of plans were not considered in the original design of the planner. In implementing TraumaTIQ, therefore, I have added two features to TraumAID's planner to enhance its ability to explain its plans.

Reasons for scheduling

Since TraumaTIQ produces comments when a scheduling constraint is being violated, it is important for it to be able to explain the reason for that constraint. As mentioned in the previous section, there are several reasons that TraumAID might schedule one action before another. In the original planner, these reasons were not saved after the scheduling constraint had been added to the plan. For the purposes of the critique, I have modified the planner so that whenever it adds a scheduling constraint to a plan it saves the reason for that constraint to be included in a comment if necessary.

Reasons for elimination of procedures

TraumAID is able to justify the actions selected by the planner by referring to the goals they address. When the planner is being used in conjunction with a system for critiquing proposed plans, it is also very important to be able to explain the reasons for *not* including certain procedures in the plan.

The second feature I have added to the planner is that when it rejects a procedure as a way of addressing a relevant goal, the reason for that procedure's rejection is recorded.

The possible reasons for not using a procedure are: contraindication, lack of resources, being in a site where the procedure cannot be done, or problems scheduling the action with respect to other actions in the plan. If one of these situations arises, the planner records it in connection with the goal being pursued. Then, if the physician orders the rejected procedure, TraumaTIQ can present a reason for not choosing that procedure to address the goal.

7.2 An Example Case

In this section, I will go step by step through an actual management plan taken from one of the 97 actual trauma cases used in the validation of TraumAID's plans (case #AP6-900463). At each point in the case, I will describe the output TraumaTIQ would produce.

The information I used for this example comes from an abstracted description of the case that was shown to the judges in the validation study. The abstracted case includes an ordered list of actions that were done, and when relevant gives their results. However, it lacks information about the timing of the case. First, there is no sense of the amount of time that passed between consecutive actions. Second, only the order in which actions were performed is recorded, not when they were ordered, so there is no way to tell when the intention to do an action first became known. Third, tests are listed in the order in which their results became available, so there is no indication of when a test was done as compared to when the results were returned. I have added some temporal information to this case for the purposes of the example.

At the start of the case we are presented with a patient with an epigastric stab wound (the epigastrium is in the center of the upper abdomen, right below the sternum or breast-bone). The initial findings (see Figure 7.2) show that the patient is not in shock or unconscious but is obtunded, meaning in a state close to unconsciousness. In addition, the evaluation of the abdomen shows no clinical signs of intra-abdominal injury, including no distended abdomen, no abdominal tenderness, and no evisceration of the abdominal contents through the stab wound.

The fact that the patient is obtunded but does not show any signs of intra-abdominal injury leads TraumAID to derive the goal of ruling out an abdominal wall injury. To

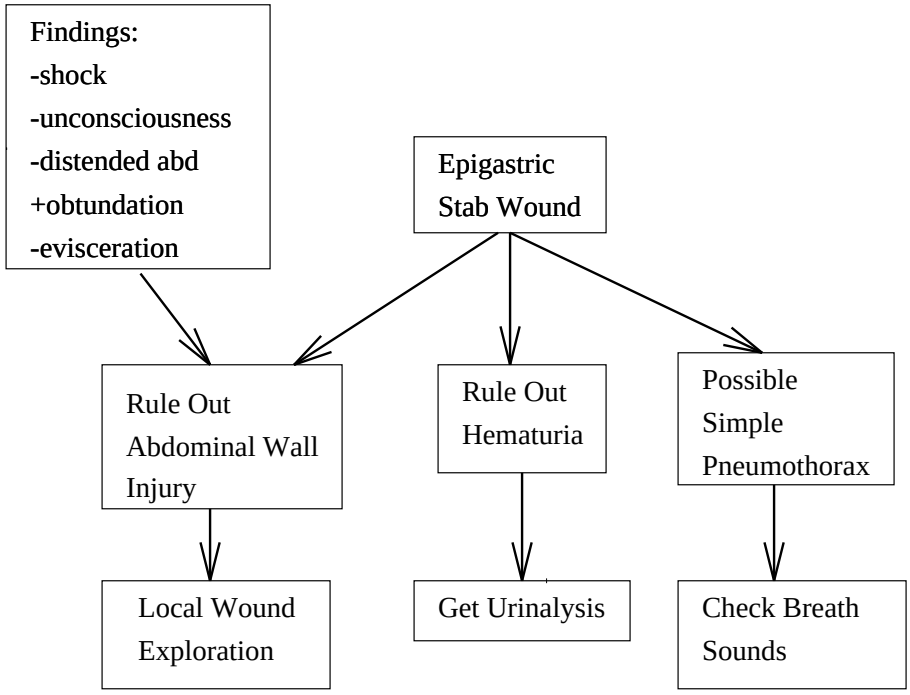

Figure 7.2: The stab wound and initial findings

address this goal, TraumAID's planner adds the action LOCAL-WOUND-EXPLORATION to its plan. A second goal, RULE-OUT-HEMATURIA is concluded on the basis of the abdominal injury, and results in the addition of a URINALYSIS to the plan. In addition, the presence of an abdominal wound triggers the suspicion of a possible simple pneumothorax (air in the chest cavity), which causes TraumAID to ask about the patient's breath sounds.[2] Figure 7.2 shows the input to TraumAID and its resulting goals and plan contents at this point.

At this point in the case, the physician ordered two actions, a naso-gastric aspiration, and a survey chest X-ray, neither of which was recommended by TraumAID on the basis of the initial information. In the plan recognition phase, TraumaTIQ infers that the naso-gastric aspiration is being done to rule out an esophageal injury, because that is the only reason it knows about for doing that action. On the other hand, there are many possible reasons for doing a survey chest X-ray, none of which are currently relevant since

[2]TraumAID's notation uses the prefix RO, for "Rule Out," to signify a diagnostic goal calling for definitive testing, and RX, for "Treat," to signify a therapeutic goal. Conclusions preceded by the word "Possible" signify that the finding is suggested by the physical evidence so far, calling for additional physical examination, but that there is not yet enough evidence to justify a definitive diagnostic test.

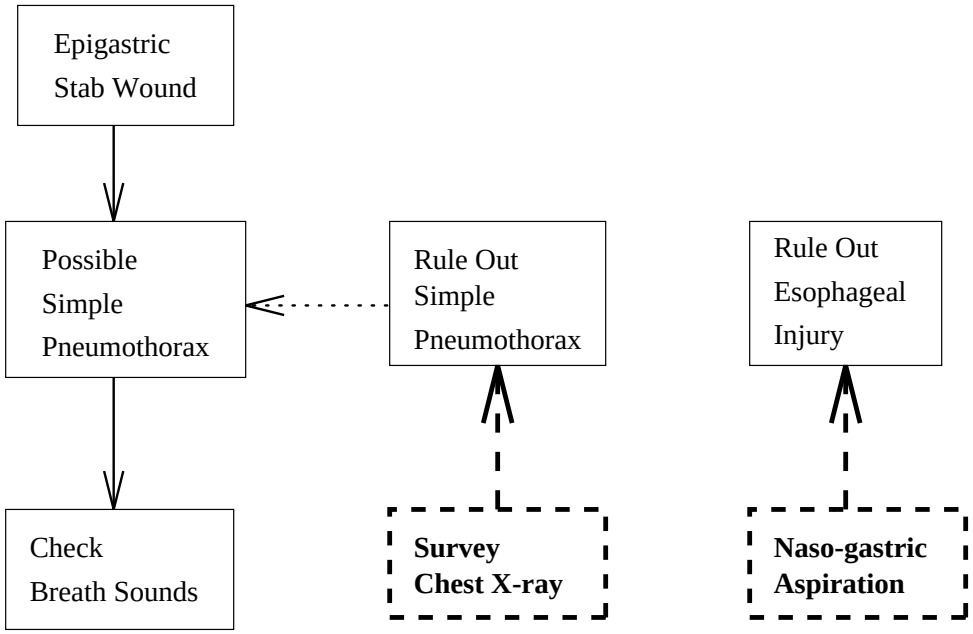

Figure 7.3: Errors of commission

the patient's wound is in the abdomen and there are no clinical signs suggesting that there might be a chest injury. The only goal that might suggest getting a chest X-ray in this situation is the remote possibility of a simple pneumothorax. TraumaTIQ therefore infers that the physician has ordered the chest X-ray to rule out a possible pneumothorax. Figure 7.3 shows TraumaTIQ's inferences, represented by upward dashed arrows in the diagram.

During plan evaluation, TraumaTIQ identifies both the naso-gastric aspiration and the chest X-ray as errors of commission, since they are in the physician's plan and not in TraumAID's plan. The naso-gastric aspiration is classified as an incorrect action because TraumAID has eliminated the goal of ruling out esophageal injury on the basis of its knowledge of the case so far. On the other hand, since the goal of ruling out a pneumothorax is potentially relevant, pending the finding of decreased breath sounds, the chest X-ray is classified as a premature action.

Both of these errors of commission, denoted by dashed boxes in Figure 7.3, are classified as non-critical errors, which results in the two comments shown in Figure 7.4 being produced in the language generation phase.

At this point, a urinalysis is done, and is removed from TraumAID's plan, as shown in

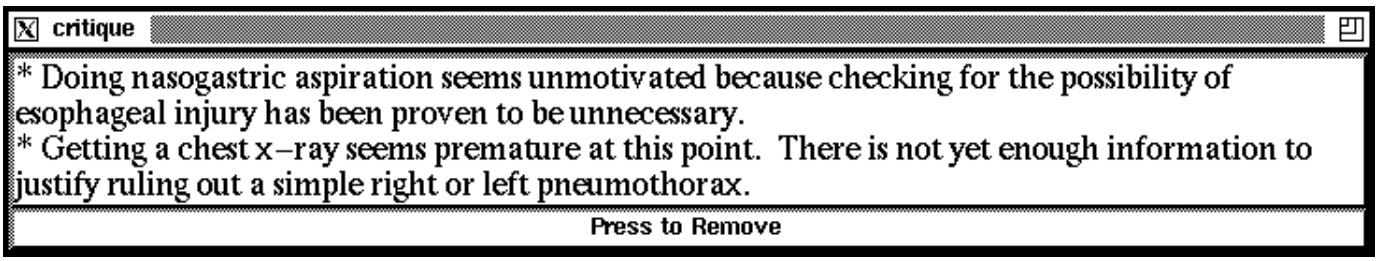

Figure 7.4: Critique for errors of commission

Figure 7.5. There are two procedures in which a urinalysis can participate in TraumAID's knowledge base: GET-URINALYSIS, which is simply done to rule out hematuria, as in Figure 7.3, and GET-PERITONEAL-LAVAGE, which can be done either to RO-SUSPICIOUS-ABDOMINAL-WALL-INJURY or to RO-ABDOMINAL-BLEEDING. Note that the first goal motivating a lavage is also relevant at this point, but TraumaTIQ infers that the urinalysis was done only to rule out hematuria, since there are several other actions in the lavage procedure that have not yet been ordered.

However, enough time has passed without the physician having addressed the goal of ruling out a suspicious abdominal wall injury, that TraumaTIQ identifies it as an error of omission. In addition, the presence of an abdominal stab wound suggests the remote possibility of a pneumothorax, which requires checking the patient for decreased breath sounds. Since this has not yet been done, it is also considered an error of omission, and the critique shown in Figure 7.6 is displayed.

The redundancy seen in the first two sentences of the critique is due to the fact that TraumaTIQ identifies errors of omission in terms of the goals that are being omitted. While these two comments involve the same action, it is aimed at two different goals – checking for a *left* or *right* pneumothorax. In Chapter 9 I will discuss the possibility of reducing this kind of redundancy by doing global sentence planning during the language generation phase.

In Figure 7.7 the physician has ordered a peritoneal lavage. At this point, the only relevant explanatory goals for the lavage action is RO-SUSPICIOUS-ABDOMINAL-WALL-INJURY, so TraumaTIQ infers that as the reason for doing the lavage. The absence of decreased breath sounds has also been reported, ruling out the suspicion of a possible pneumothorax.

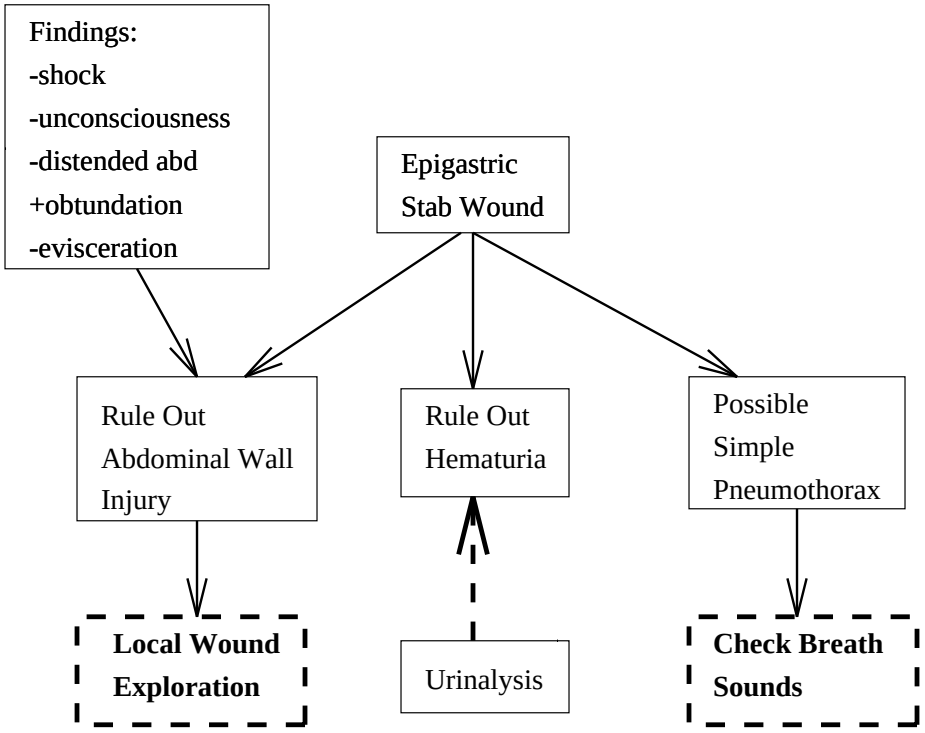

Figure 7.5: Errors of omission

During plan evaluation, TraumaTIQ notices that this goal is being addressed by different procedures in TraumAID's plan (local wound exploration) and in the physician's plan (peritoneal lavage). It identifies this discrepancy as a procedure choice error.

Even though the lavage is not TraumAID's preferred procedure for ruling out abdominal wall injury, the plan evaluator notes a scheduling error resulting from the fact that the physician has not yet checked for abdominal scarring, which is a precondition for doing lavage. This will be commented on in the critique.

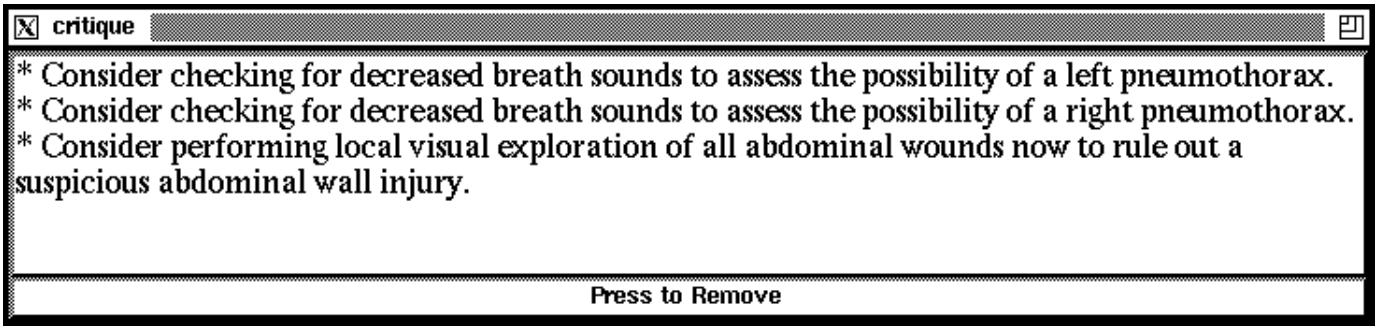

Figure 7.6: Critique for errors of omission

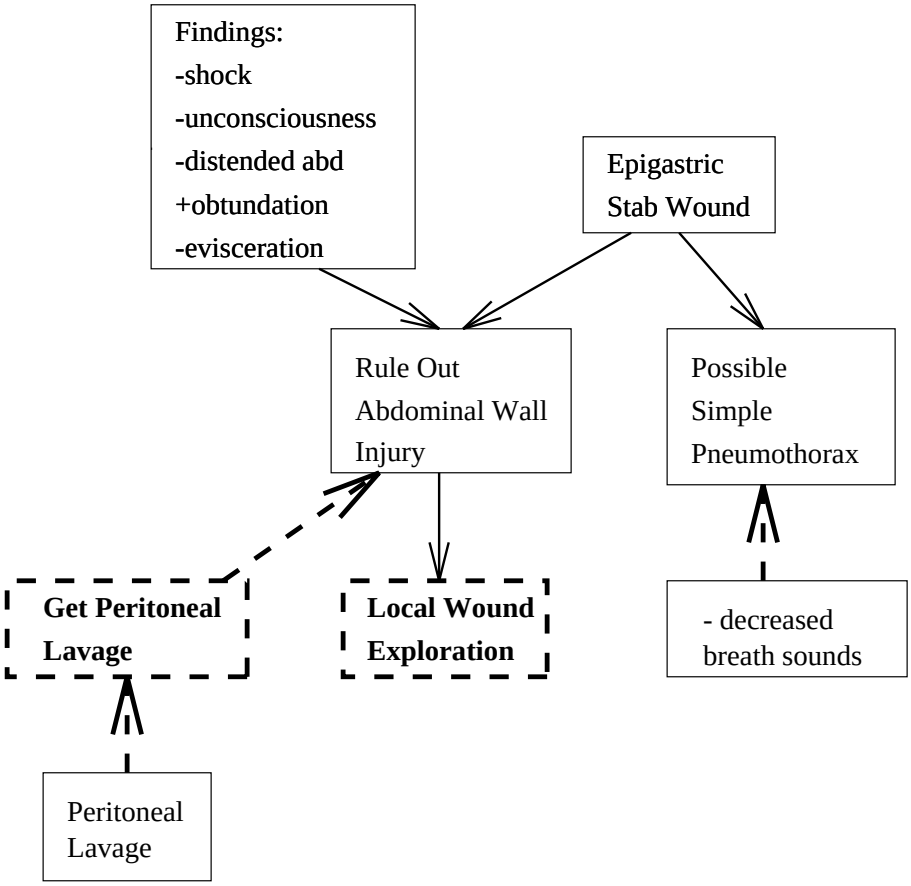

Figure 7.7: Procedure choice error

On the other hand, since the physician is now addressing the goal RO-SUSPICIOUS-ABDOMINAL-WALL-INJURY that was the topic of the earlier error of omission comment, this error is no longer present in the physician's plan, and so the comment is not repeated in the resulting critique, shown in Figure 7.8.

In the next stage of the case, Figure 7.9, the lavage is still pending.[3] The physician now orders observation of the patient for abdominal bleeding or peritonitis, an action that would be taken in case of a diagnosed abdominal wall injury.

Since the results of the lavage are not yet available, the diagnosis of abdominal wall injury is not yet justified. However, because the goal of treating an abdominal wall injury is a later stage of the diagnostic strategy currently being pursued by doing a peritoneal lavage, the action of observing the patient is interpreted as a premature action, rather than

[3]Since the physician managing this case did not have access to TraumaTIQ's comments, I assume that the original order for a lavage remains after it has been critiqued as a procedure-choice error.

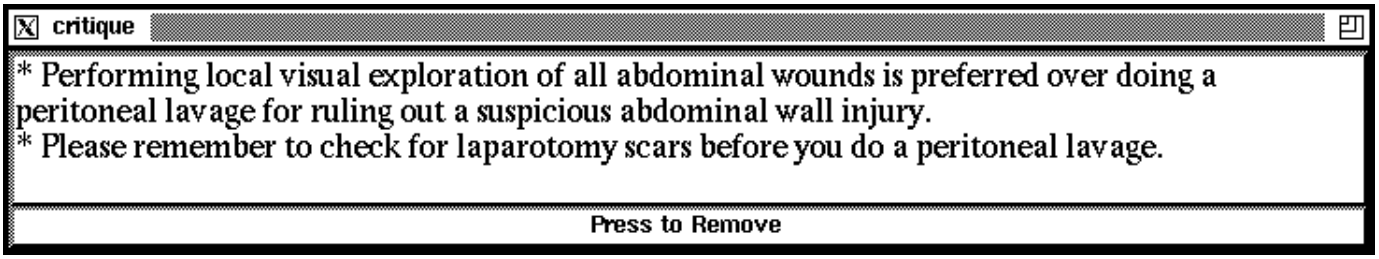

Figure 7.8: Critique for procedure choice error

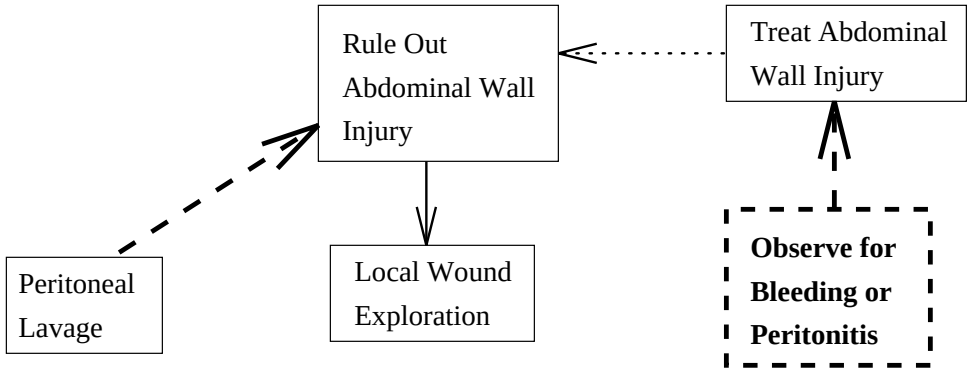

Figure 7.9: Premature Action

being completely unmotivated. This results in the critique shown in Figure 7.10.

After a period of time the results of the peritoneal lavage, which was negative, are entered. The goal of treating an abdominal wall injury is now no longer potentially relevant, so the pending action of observing the patient for bleeding or peritonitis is no longer explainable in terms of a unique goal. In addition, since the patient remains obtunded with no signs of intra-abdominal injury, TraumAID derives a goal of retaining the catheter

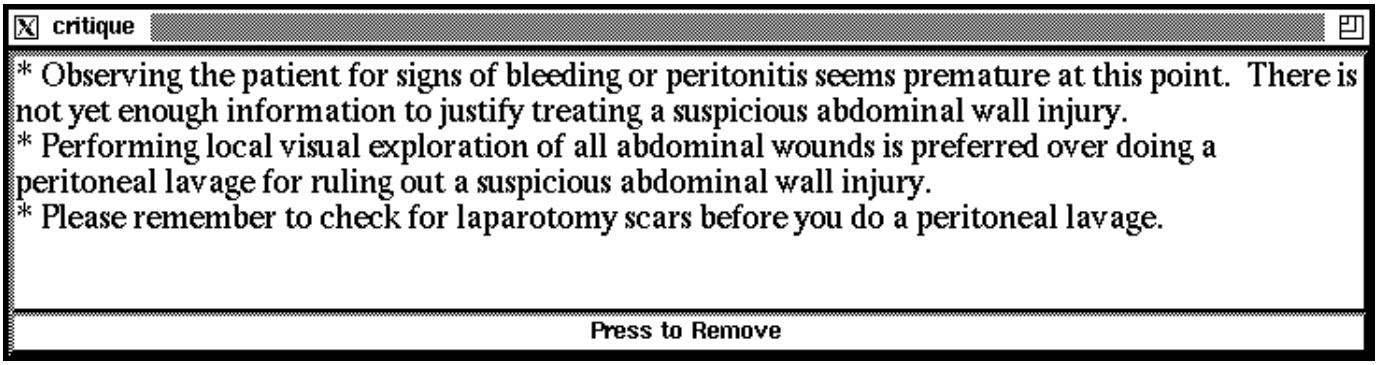

Figure 7.10: Critique for premature action

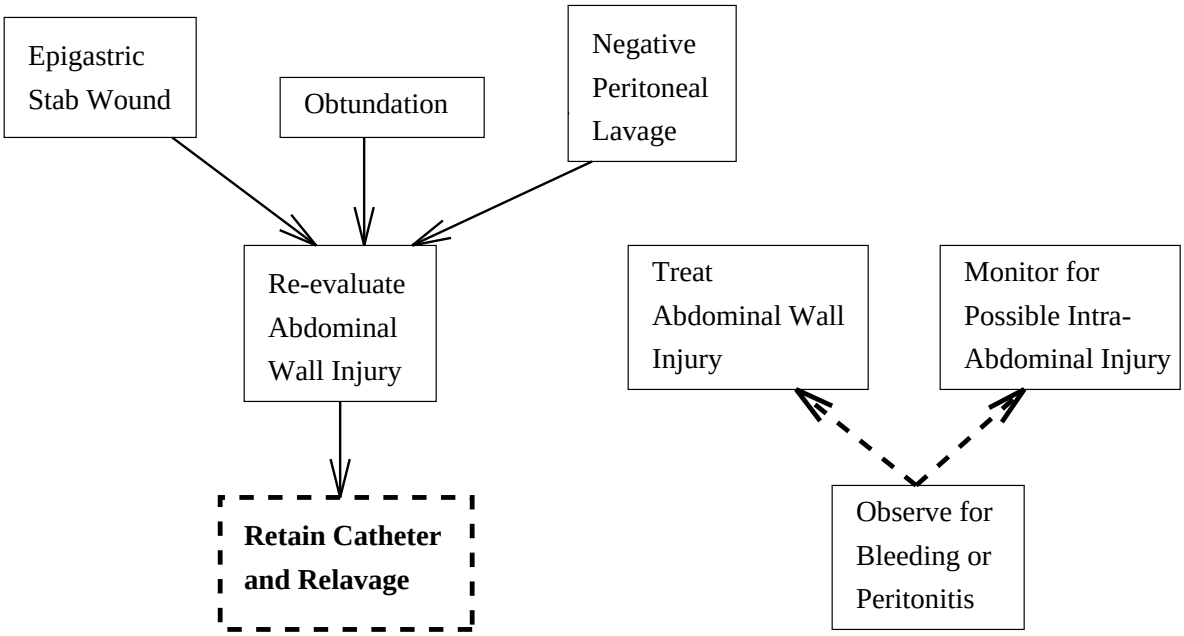

Figure 7.11: Critical error of omission

that was used for the lavage and relavaging after 6 hours. The plan at this point is shown in Figure 7.11.

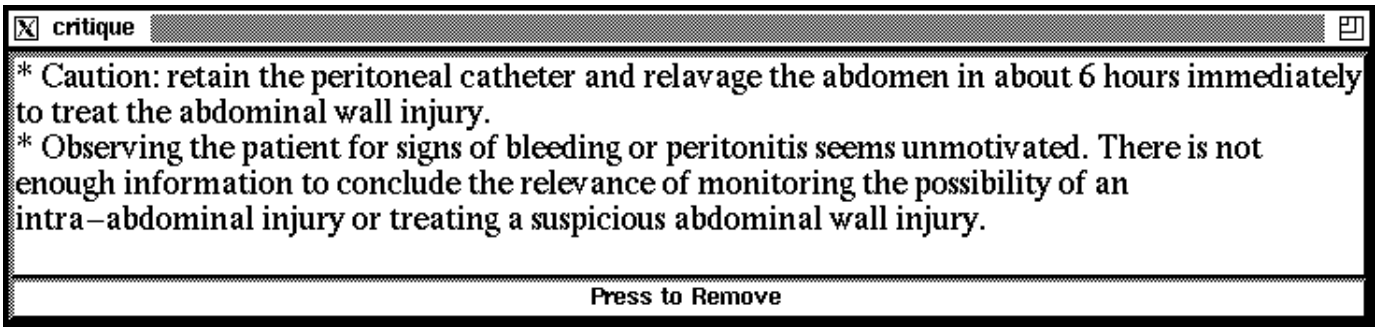

Figure 7.12: Critique for critical error of omission

During plan evaluation, TraumaTIQ notes the omission of the relavage procedure, which it classifies as a critical error. It also classifies the pending observation for bleeding or peritonitis as an unjustified error of commission. These errors result in the critique shown in Figure 7.12. The word "immediately" in the first comment is part of the template for critical errors of omission, which was added to convey the seriousness of the comment. This is a good example of the drawbacks of using templates to generate output since in this case it results in the confusing juxtaposition of "immediately" and "in about 6 hours."

At this point the record of the case ends.

Chapter 8

Evaluation

Before putting a decision support system into use it is important to have some evidence that its output is correct and has the potential to improve performance. In the past, very few developers of critiquing systems did any kind of evaluation of the output of their systems, with the notable exception of Van der Lei, who carried out an extensive evaluation of his HyperCritic system [90].

There are a number of factors by which to evaluate a critiquing system.

1. Correctness: Are the critiques produced by the system correct?
2. Clinical significance: Do all the critiques generated by the system have the potential to improve patient outcome?
3. Completeness: Does the system produce all the critiques warranted by the observed behavior?
4. Timeliness: Is the critique generated quickly?
5. Means of communication: Are the critiques conveyed to the user in an acceptable way, with an appropriate amount of explanation to back them up?
6. Effectiveness: Does the system actually influence behavior and/or outcomes?

These questions can be divided into two categories: the first four can in part be answered off-line, before the system is experienced by real users in the task it is designed

for, while the fifth and sixth must be answered through realistic field experience with the system.

8.1 Field testing of TraumAID and TraumaTIQ

The questions regarding the effectiveness of TraumAID in real trauma situations will be investigated in a future study in which TraumAID and TraumaTIQ will be installed in a trauma bay at the Medical College of Pennsylvania. This will involve a comparison of different modes of critique output, and observation of the users' interaction with the system. As people have started to tackle the problem of having decision-support systems accepted into routine clinical use, the importance of field testing has become clear [26, 27, 95]. This type of evaluation poses a number of challenges, some of which are discussed in [95]:

- Representativeness of the setting: The setting in which the system is evaluated may not be representative of the intended setting and/or users.

- Measurement of effectiveness: Ideally, the question of interest is whether the system has a positive effect on patient outcome. But this may be much more difficult to measure than other items, such as its effect on physician decisions or behavior.

- Design of randomized trials: Care should be taken in the planning of controlled trials. In evaluating a decision-support system, it may be necessary to randomize not only patients, but also doctors or even hospitals.

- Correcting for biases caused by the trial: The fact that an experimental trial is being conducted may have certain biasing effects on physician behavior. For example, they may try to allocate certain cases to the decision-support group, their overall performance may improve merely as a result of being studied. They may respond negatively to the system if participating in the trial requires them to do extra work. Another form of bias may occur on the part of the people assessing the outcome of the experiment, particularly if they have preconceptions about the outcome.

- Correcting for biases due to the decision aid: The fact of using a decision-support system may influence performance in other ways than the intended effect. For example, if the system has a tutorial effect it may carry over to cases where the system is not used. In addition, physicians using the system may be motivated to perform better because their behavior is being more closely scrutinized.

- Cases where advice is not used: All cases where the physician was given the opportunity to use the decision-support system should be included in the analysis, even if the system was not actually used, or if the advice was ignored.

- The evaluation paradox: Physicians may be reluctant to follow the advice of a system that has not been shown to improve decisions, but this improvement cannot be shown unless the advice is followed. Therefore, it is important to explain to physicians participating in the trial the reasoning process used by the system as well as its performance in retrospective evaluations. In addition, systems that back up their advice with explanations are more likely to be followed [60, 86].

These problems will need to be addressed in the prospective evaluation of TraumAID 2.0 which 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, TraumAID'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.

Following this evaluation of the effectiveness of displaying TraumAID's plans, the critiques generated by TraumaTIQ will be introduced. In order to evaluate the potential advantages (or disadvantages) of employing a critiquing approach for conveying decision support for trauma management, the physicians' reactions and behavior in response to to the critiquing mode will be compared to their response to the graphical display of plans.

This proposed evaluation will also compare two modes of critique delivery to the physician: directly via synthesized speech, or indirectly via displaying the critique in text form to the scribe nurse who can then convey its messages to the physician. The design of this study is motivated by the understanding that interpersonal interaction and relative status

of members of the trauma team will play an important role in TraumAID's acceptance. The effect of adding a computer to the trauma team cannot fully be anticipated in advance, and it is important for the system to be able to assimilate into the social milieu of the trauma bay.

8.2 Retrospective evaluation of TraumaTIQ

While this proposed field study of TraumAID and TraumaTIQ will provide invaluable information regarding the communicative needs of physicians in the trauma situation, as a preliminary evaluation of TraumaTIQ I have concentrated on the questions that can be answered off-line, before real experience with the system is possible. To do this, I made use of data from the original validation study of TraumAID, which was mentioned in the Introduction to this dissertation.

In the validation study, three trauma experts were given abstracted descriptions of 97 actual trauma cases involving penetrating injuries to the chest and abdomen that presented consecutively at the Medical College of Pennsylvania. They were also given descriptions of the management plan that TraumAID would have produced for those 97 cases. The judges performed a blinded comparison of the two management plans in which they were asked to give each plan a rating on a scale of one to four, with one being completely unacceptable, and four being completely acceptable. When both plans were given the same rating, they were asked to note which they preferred. The results of this study were that TraumAID's plans were preferred over the actual care to a statistically significant extent.

During the validation study, the judges were also asked to identify the individual errors of omission, commission, and scheduling that occured in each case. This information was not used for the validation study, but we noticed that these comments were comparable to the output of TraumaTIQ, and thus could be used as a source of data for evaluating the comments produced during critiquing. It is important to remember, however, that the judges comments were not produced during observation of the cases, and were not their primary task. Thus they are likely to be incomplete and, occasionally, inaccurate.

To evaluate TraumaTIQ, I ran it in batch mode on the descriptions of these 97 cases, resulting in a list of critique comments for each case. The case descriptions were in the form

of lisp-readable lists of actions with their corresponding results when applicable. Figure 8.1 shows an example of one of these case descriptions. TraumaTIQ's batch mode differs from normal operation in the following ways:

- The order of actions in the case corresponds to the order in which they were done. There is no indication of when they were ordered, how much time passed between actions, or where they were performed.

  To critique these cases, TraumaTIQ processed them as if each action was ordered just before it was done, and no other actions were ordered between the ordering of an action and its performance. This means that the effect on the critique of ordering several actions at a time would not be captured.

- In a real case, TraumaTIQ waits to comment on errors of omission until a certain period of time has passed. As described in Section 5.3, the amount of time is determined by the urgency of the omitted goal: *catastrophic* goals are mentioned immediately, *unstable* goals are mentioned after 2 minutes, and *stable* goals are mentioned after 5 minutes.

  Lacking temporal information from the cases, errors of omission regarding actions with the second and third levels of urgency were not commented on until the entire case had been processed without observing that goal to be addressed.

- When TraumaTIQ critiques an error of omission it often groups several actions together into one comment about failure to address a goal. If some of these actions are subsequently done, another comment will be produced regarding the remaining actions. The information in the earlier comment *subsumes* the later comment since it contains all the same information. In batch mode, any comments that were subsumed by other comments were removed from the output.

- Repeated comments were removed from the output.

The result of running TraumaTIQ on a case in this manner was a list of unique comments regarding the management presented in the case.

In addition to TraumaTIQ's output on these cases, the evaluation also made use of the judges' comments from the original validation study of TraumAID [18]. In that study,

```
;; This is the ACTUAL case from the EXCEL file.
;; Data is from the file: #P"/tmp/excel-files/AP6-890613.excel"

;; ------------------------------------------------------
;; The following are the INITIAL (ord = 0) findings:

(CASE_FORMAT "Validation Format Two")

(CASE_INFO (CASE_RECORD_ID . "X2rap6-890613") (AGE . 19) (SEX . "M")
 (COMPLICATIONS . "None") (SURVIVED . T) (DISABILITY))
;; ------------------------------------------------------

(INITIAL_FINDINGS
 (("Wound" (WOUND-TYPE . GUNSHOT)
   (WOUND-LOCATION . "Left_Lumbar_Posterior")
   (WOUND-DIRECTION . UNKNOWN) (SIDE . LEFT)
   (LATERAL . TRUE) (NUMBER . 1))
  . POSITIVE_FACT)
 "No other wounds" (("Shock") . NEGATIVE_FACT)
 (("Unconsciousness") . NEGATIVE_FACT) (("Obtundation") . NEGATIVE_FACT)
 (("Decreased_Breath_Sounds" (SIDE . LEFT)) . NEGATIVE_FACT)
 (("Decreased_Breath_Sounds" (SIDE . RIGHT)) . NEGATIVE_FACT)
 (("Muffled_Heart_Sounds") . NEGATIVE_FACT)
 (("Loss_Motor_Leg" (SIDE . RIGHT)) . NEGATIVE_FACT)
 (("Loss_Motor_Leg" (SIDE . LEFT)) . NEGATIVE_FACT)
 (("Urinalysis_Rbc" (TEST-RESULT . NEGATIVE)) . POSITIVE_FACT))
;; ------------------------------------------------------

(REST_OF_FINDINGS

(("Distended_Abd") . NO)
(("Tenderness") . NO)
(("Check_For_Laparotomy_Scar" (TEST-RESULT . POSITIVE)) . NO)
(("Loss_Sensation_Leg" (SIDE . LEFT)) . NO)
(("Absent_Rectal_Tone") . NO)
((TEST-INTERPRETATIONS-MENU ("Survey_Chest_X_Ray" NIL) NIL T T)
 ((TEST-INTERPRETATIONS-MENU ("X_Ray_AP_Abd" NIL) NIL NIL T)
  ("X_Ray_Bullet_Abd_Cavity_not_Midline" (TEST-RESULT . POSITIVE))
(("Bullet_Image" (AP-LOCATION . "Abd_Cavity_Not_Midline")
   (LAT-LOCATION . "Abd_Cavity_LAT") (BULLET-ID . 1))
  . POSITIVE_FACT)
((TEST-INTERPRETATIONS-MENU ("X_Ray_Lat_Abd" NIL) NIL NIL T)
  ("X_Ray_Bullet_over_Abd_Cavity" (TEST-RESULT . POSITIVE)))
(("Observation_For_Bleeding_Or_Peritonitis") . YES))
```

Figure 8.1: A TraumAID-readable actual case description

the judges were given text versions of the actual management plans and the management plans that would have been generated by TraumAID 2.0 on the same cases, and did a blinded comparison of the two. For each case, the judges were asked to record errors of commission, omission and scheduling, and to give the management an overall rating on a scale of 1-4, with 4 being perfectly acceptable with no errors, and 1 being unacceptable.

8.3 Results

TraumaTIQ's output on the 97 actual cases is summarized in Table 8.1. The comment types listed in the table correspond to the error classes enumerated in Chapter 5. Each cell contains the total number of comments on the 97 cases of the corresponding type and significance level.

The action of checking for medication allergies was not reported in the abstracted case records and so was not involved in any of the errors noted by the judges. This action's absence was responsible for 44 errors of omission and 109 precondition scheduling errors noted by TraumaTIQ. In the remainder of this evaluation, comments having to do with checking for medication allergies have been eliminated from consideration.

8.3.1 Correctness

The correctness of a critique involves both the correctness of its advice and the correctness of its inferences. In TraumaTIQ, the correctness of the system's advice (what it presents as the recommended course of action) is a matter of the correctness of TraumAID's plans since its recommendations are based directly on those plans. As previously described in Chapter 1, the management plans produced by TraumAID have been judged to be acceptable by a panel of experts (cf. [75]).

8.3.2 Clinical significance

Another dimension on which TraumaTIQ's output can be evaluated is the sensitivity of its judgments of the clinical significance of errors. The evaluation of the errors identified during plan evaluation produces a measure of disutility for each comment. By combining these individual disutilities we can get a measure of the overall disutility of the case according to

	Comment Level		
Comment Type	Warn	Inform	Ignore
Errors of Omission			
Goal partially omitted	13	10	50
Goal completely omitted	126	38	0
Bedside questions omitted	236	118	0
Errors of Commission			
Unmotivated action	10	13	33
Premature action	4	41	14
Erroneous action	24	61	6
Redundant action	0	0	0
Prohibited action	0	2	0
Procedure Choice Errors			
Prohibited action	1	0	0
Preferred action	0	12	0
Optimized action	1	0	0
Scheduling Errors			
Urgency	0	14	0
Priority	0	45	0
Site	3	14	36
Precondition	0	111	0
Precedence constraint	0	4	0
Informational dependency	0	5	8

Table 8.1: Comments per case produced by TraumaTIQ on actual cases

TraumaTIQ. In this analysis I have used two different ways of combining disutilities and compared the resulting numbers to the overall case ratings of the three judges.

The first combination function I tested was the sum of the disutilities for all the comments made in the case. Looking at the resulting numbers shows that the mean total disutility for the 97 cases is 104.7, with a standard deviation of 68.9. The maximum case disutility is 389, the minimum is 1, and the median total disutility is 89. In 43 out of the 97 cases, the total disutility is greater than 100, meaning that the disutilities of all errors in management added up to an experience worse than the worst possible single experience, which was taken to be dying. This is an extremely harsh judgment, but if the management was not really that bad, then what is going on? The answer is that by summing the disutilities we are treating them as if they are both independent and cumulative, when in fact they are clearly not. For example, the disutility of failing to repair an injured kidney *and*

failing to check for abdominal tenderness can easily be seen to be less than the sum of the disutilities of the two errors since once a kidney injury has been diagnosed the abdominal tenderness becomes irrelevant.

The second approach I took, therefore, is based on the idea that "a chain is only as strong as its weakest link." In this model I simply take the error with the highest disutility in each case to represent the overall disutility of the case. This makes the opposite assumption of the previous approach – that the errors are all dependent on each other and more serious errors subsume less serious errors – and thus provides a lower bound for the case disutility. Using this measure for case disutility, the mean value is 32.8 (s.d. 18.31). The maximum is 95, the minimum is 1, and the mean is 30.

We are interested in how well both disutility measures predict the overall evaluation of cases by human experts. Table 8.2 shows the results of a regression analysis of the relationship between the sum of TraumaTIQ's comment disutilities and the judges' case ratings.[1] As Table 8.2 shows, TraumaTIQ's total comment disutility is a significant predictor of the ratings given to cases by Judges 1 and 2, but not for Judge 3 ($p = 0.21$).

Table 8.3 shows the results of regressing TraumaTIQ's maximum comment disutility value on the judges' case ratings. The results in Table 8.3 show that the maximum disutility is a significant predictor of all three judges ratings.

In both of these regressions, the low values of the adjusted R^2, which indicates the amount of variance in the judges ratings that is explained by the model (the total comment disutility in Table 8.2 and the maximum comment disutility in Table 8.3), says that the fit between the dependent and independent variables is weak. The fact that the adjusted R^2 value for Judges 1 and 2 goes down from the first model to the second suggests that the sum of disutilities is a slightly better model of those two judges overall case rating than the maximum disutility. On the other hand, Judge 3 appears to prefer to judge a case on the basis of its most egregious error. It appears that considering the disutilities of errors somewhere between their individual and cummulative maximums has some correlation with the judges' ratings. Consistent with estimates of disutility, different judges may assess these overall disutilities differently.

[1] The dependent variable in each model is the judges' rating for the individual cases. Each model is estimated on 97 cases. *** $p < .01$, ** $p < .05$

Variable	Judge 1	Judge 2	Judge 3
Intercept	2.98***	3.002***	2.441***
	(.184)	(.137)	(.192)
TraumaTIQ cost	-0.006***	-0.004***	-0.002
	(.001)	(.001)	(.002)
adjusted R^2	.13	.12	.01

Table 8.2: Models of Judges' Ratings using TraumaTIQ's total comment disutility.

Variable	Judge 1	Judge 2	Judge 3
Intercept	3.02***	2.957***	2.694***
	(.210)	(.160)	(.212)
TraumaTIQ cost	-0.019***	-0.012***	-0.014**
	(.005)	(.004)	(.006)
adjusted R^2	.10	.07	.05

Table 8.3: Models of Judges' Ratings using TraumaTIQ's maximum comment disutility.

The judges seem to be quite different in what factors they use to evaluate management plans. Looking at the judges' ratings in more detail, we see that the correlation between the three judges is quite low, as shown in Table 8.4. There is only a significant positive correlation between Judges 1 and 3 and between Judges 2 and 3.

	Judge 1	Judge 2	Judge 3
Judge 1	1		
	(0.0)		
Judge 2	0.0870	1	
	(0.397)	(0.0)	
Judge 3	0.322	0.215	1
	(0.0013)	(0.034)	(0.0)

Table 8.4: Pearson Correlation Coefficients / $Prob > |R|$ under H_0: $\rho = 0$ / $N = 97$

It is also interesting to look at the occurrence of individual comment types, such as the number of errors of omission and commission, to understand the characteristics of a case that are predictive of the judges' ratings. In order to gain a better understanding of the relationship between the judges' ratings and the errors they noted in the cases we estimated

Variable	Judge 1	Judge 2	Judge 3
Intercept	2.990*** (.126)	3.022*** (.108)	2.750*** (.169)
Number of Errors of Commission	-.184** (.081)	-.305*** (.081)	-.144 (.156)
Number of Errors of Omission	-.327*** (.047)	-.154*** (.033)	-.199*** (.045)
adjusted R^2	.34	.24	.16
p value for Difference Test	.12	.07	.72

Table 8.5: Models of Judges' Ratings using Judges' reported numbers of errors of omission and errors of commission.

a set of regression models. These models regressed the individual judges' overall rating of each case on the number of errors of omission and the number of errors of commission which the judge attributed to each plan. The results of these models appear in Table 8.5.

In evaluating the results of the models presented in Table 8.5,[2] there are a number of different factors to consider. First, we are interested in the statistical significance and magnitude of the parameter estimates for the number of each type of error (numbers of errors of commission and omission) on the judges' ratings. We also want to test the hypothesis that the marginal effect of an error of commission is different from the marginal effect of an error of omission for each of our three judges. Finally, we are interested in the overall goodness of fit for each of our models of Judges Ratings.

From the parameter estimates for the impact of the number of each type of error on overall judge ratings, we see that the number of errors of omission is a statistically significantly predictor of the overall rating given to plans by all three judges. The number of errors of commission is a statistically significant predictor of overall ratings for Judges 1 and 2.

For each of these models we tested the hypothesis that the marginal effect of an error of omission is different from the marginal effect of an error of commission. The p value for each of these tests is presented at the bottom of Table 1. We are only able to reject the null hypothesis that there is no difference in the marginal effect for the two different types

[2]The dependent variable in each model is the judges' rating for the individual cases. Each model is estimated on 97 cases. *** $p < .01$, ** $p < .05$

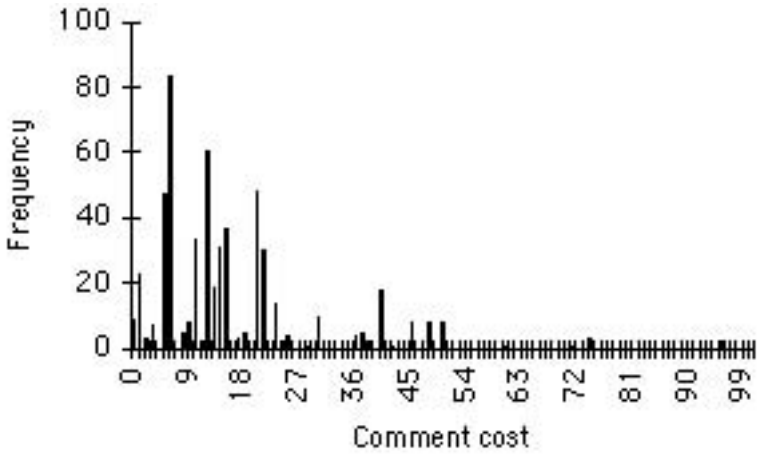

Figure 8.2: Frequency of disutilities for errors of omission

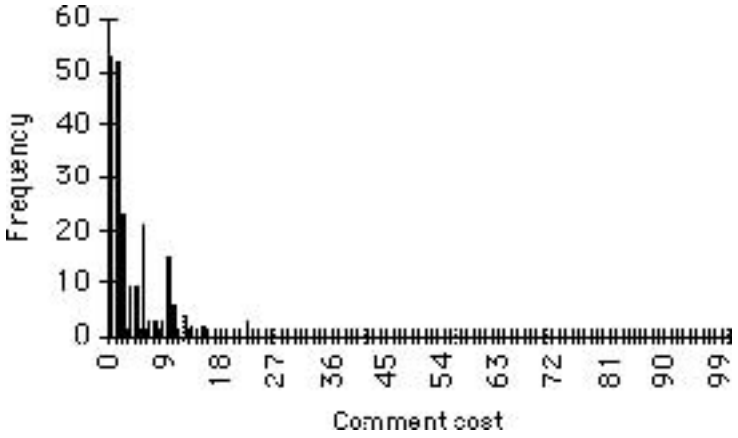

Figure 8.3: Frequency of disutilities for errors of commission

of errors in the model for Judge 2, in which the parameter value for errors of commission is greater than for errors of omission.

The distribution of disutilities for errors of omission and errors of commission are shown in Figures 8.2 and 8.3. TraumaTIQ tends to assign higher disutilities to errors of omission, (mean = 15.85) than it does to errors of commission (mean = 3.33) In addition it identifies many more errors of omission (5.64 per case) than errors of commission (2.14 per case). Since the maximum possible disutility for an error of commission is 65 and the maximum possible disutility for an error of omission is 95, this suggests that physician's tend not to do costly actions unnecessarily, while they are more likely to commit costly errors of omission. The focus of TraumaTIQ's critiques, therefore, will be more on reminding

		0	1	2	3
All comments	TraumaTIQ yes	345	71	21	5
	TraumaTIQ no	NA	76	10	1
Commission	TraumaTIQ yes	159	42	13	5
	TraumaTIQ no	NA	30	2	0
Omission	TraumaTIQ yes	186	29	8	0
	TraumaTIQ no	NA	46	8	1

Table 8.6: Comment-by-comment agreement between TraumaTIQ and judges on actions

physicians about actions that they *should* be performing, rather than on informing them of actions they *should not* perform.

8.3.3 Completeness

The completeness of a critique is a measure of whether the critique includes every item that it should. One way to evaluate this is to compare the critiques produced by TraumaTIQ to the comments of physicians on the same management plans. Table 8.6 shows the results of a comment-by-comment comparison of TraumaTIQ's critiques on the 97 cases with the comments made by the three local judges on the same cases.

This table includes only comments on actions, not bedside questions, since TraumaTIQ does not comment on errors of commission involving bedside questions. TraumaTIQ's comments on errors of omission involving more than one action have been broken up into individual comments for each action. Comments regarding omission of checking for medication allergies were not included because this action was not included in the abstracted case descriptions.

I excluded comments regarding scheduling errors from this evaluation because TraumaTIQ does not comment on the relative order of two actions that have both been done, which is what the judges did. Rather, TraumaTIQ's scheduling comments are designed to remind the physician of an ordering constraint if it seems that she is going to do the second action without having ordered the first. Since there is no information in the abstracted cases about when, or in what order actions were ordered, it is impossible to evaluate TraumaTIQ's scheduling comments in this way.

The columns in Table 8.6 correspond to the number of judges making a particular comment. It has been demonstrated [18, 90] that there is often little agreement between physicians on what constitutes an error that should be commented on. I therefore hypothesized that the greater the agreement between judges on an individual comment, the stronger the evidence that that comment should be included in the critique.

The first two rows of the table show the comment by comment agreement of TraumaTIQ with the judges on errors of omission and errors of commission. The first column shows that TraumaTIQ made 345 comments which were not duplicated by any judge. Possible reasons for this disparity include:

- Commenting on each individual action was not the experts' primary task in the validation study, which was concerned with the overall rating of the case. The judges tended to mark individual items sporadically, sometimes only marking one error on a case that they rated as unacceptable or acceptable with major reservations. Had they been asked to mark down every comment as if they were observing the case being managed, they may have produced more comments.

- The experts tended to make one high level summary comment on the conduct of care, while TraumaTIQ fills in all the details. For example, a judge might comment that the central action of a procedure, such as a tube thoracostomy, was omitted without mentioning the other actions that should be done before or after that action, such giving antibiotics or doing a post-tube X-ray to evaluate the position of the tube. TraumaTIQ, on the other hand, would list every action in the procedure that had not been done.

Columns 2-4 show that TraumaTIQ produced 70.3% of comments made by 2 or more judges and did not produce 51.7% of comments made by only one judge. This crossover effect is significant by chi-square ($\chi^2 = 6.215, df = 2, p < 0.05$). This result indicates a correlation between the importance of a comment (as measured by the number of judges that made it), and the likelihood that TraumaTIQ will produce that comment.

The rest of the table shows the same comments divided into errors of commission and errors of omission. This split shows that the crossover effect is much stronger for errors of

commission than for errors of omission. Even when two or more judges agree on an error of omission, TraumaTIQ only produces that comment about half the time. In fact, the correlation between the number of judges producing a comment and whether TraumaTIQ produced the comment is significant for errors of commission ($\chi^2 = 7.213, df = 2, p < 0.05$) but not for errors of omission ($\chi^2 = 1.385, df = 2, p > 0.50$). This observation suggests that TraumaTIQ is more often in agreement with experts about errors of commission than about errors of omission, an effect that can be explained by the fact that comments on errors of commission are constrained to be about actions that were *done*, but comments on errors of omission can be about any action that was *not done* – a much larger set.

8.3.4 Discussion

One useful aspect of this evaluation is that it allows us to look at the cases in which TraumaTIQ does not agree with the judges: both when it fails to produce a comment that was made by more than one judge, and when it frequently produces comments that are not produced by any judge.

In the category of comments that were made by more than one judge that were not made by TraumaTIQ, it is most often the case (7 out of 11 times) that the discrepancy arises from TraumAID having a more cautious diagnostic strategy than the physicians. In two of the cases, the judges note that an operation was omitted (one thoracotomy, one laparotomy), while TraumaTIQ comments that an assessment of an inconsistent number of bullets and bullet holes has been omitted. TraumAID will not go on to call for an operation until the bullet hole assessment has been done.

In three cases, the judges note an omission of actions related to doing a laparotomy, while TraumaTIQ thinks that the operation is done prematurely and a peritoneal lavage has been omitted. In two cases, the judges note an error of commission of a peritoneal lavage, while in contrast TraumaTIQ agrees with the lavage but considers the operation that followed an error of commission.

The other four disagreements are:

1. two judges say removal of a clotted hemothorax was omitted, which involves an operation, while TraumAID does not recommend operating unless the hemothorax

is persistent.

2. Two of the judges say a chest X-ray was omitted in a case of an abdominal wound with no findings in the chest. TraumAID does not recommend getting a chest X-ray in such a situation.

3. Two of the judges say a naso-gastric aspiration was omitted in a case where it was in fact done, and in which TraumaTIQ says it was an error of commission.

4. Two of the judges say a urinalysis was omitted in a case with a stab wound to the upper posterior chest. TraumAID would not call for a urinalysis in such a case, because the chances of abdominal injury are very low.

None of these discrepancies prompted changes in TraumAID's reasoner or planner. Rather, we conclude that they represent philosophical differences between the judges, who seem to favor a more decisive approach, and the more cautious strategy we have undertaken to encode in TraumAID.

Looking at the comments that were made by TraumaTIQ but not by the judges might also suggest possible extensions or changes to the knowledge base. For example, TraumaTIQ evaluates 23 orders for naso-gastric aspiration as errors of commission. The only goal in its knowledge base connected to that action is ruling out an esophageal injury. Since the cost of doing a naso-gastric aspiration is above the threshold for comment, it often appears in the critique with the explanation that ruling out an esophageal injury is not motivated. However, it turns out that (1) naso-gastric aspiration is often done for another reason, (namely to remove the contents of the stomach prior to doing a peritoneal lavage), although this procedure is not considered necessary by TraumAID and thus does not appear in its knowledge base, and (2) performing an unnecessary naso-gastric aspiration may not actually be significant enough to comment on. In response to this observation, it may be necessary to include additional goals in TraumAID's knowledge base to explain naso-gastric aspiration, and/or to lower the cost of the naso-gastric aspiration action.

8.4 Timeliness of the critique

A final area in which to evaluate the implementation of TraumaTIQ is how fast it generates its critiques. In the current architecture the computational bottlenecks are plan recognition, plan generation, and calculating the informational dependencies between actions. I have discussed the complexity of the plan recognition algorithm in Chapter 4.

Rymon [75] has presented an algorithm for Progressive Horizon Planning that he proves to run in polynomial time assuming that plan sketching and optimization can both be done quickly. In practice, however, this assumption depends on optimizing no more than one step in the plan, which we have found to be too great a restriction. In the current implementation, TraumAID's planner optimizes the first five goals in the plan. Fortunately, working in batch mode on the 97 cases in this evaluation, the planner only has to plan for an average of 3.24 goals at a time, and takes an average of .2 cpu seconds on a Sun 4 to generate a new plan.

The total time taken by TraumaTIQ's plan recognition, plan evaluation, and critique generation components when run in batch mode is an average of .124 cpu seconds per critiquing cycle.

Checking the dependencies between actions in TraumAID's plan is exponential in the number of actions in the plan. In this evaluation there was an average of 3.08 actions per plan and dependency checking took an average of .267 cpu seconds per planning cycle. Therefore, in the average case when the system needs to generate a new plan, check dependencies, and generate a critique, it will take just a little more than half a second.

8.5 Summary of the evaluation

In this chapter, I have presented a retrospective evaluation of TraumaTIQ that considers the issues of *correctness*, *clinical significance*, *completeness* and *timeliness* of the generated critiques.

As it depends on the correctness of the management plans generated by TraumAID, the correctness of the critiques is implied by the approval of TraumAID's plans by the expert judges in [18]. The ability of TraumaTIQ to generate clinically significant comments is

supported by the fact that its case disutility ratings correlate significantly with the ratings of two out of the three local judges. The completeness of TraumaTIQ was evaluated by looking at the agreement between the system and three expert judges on individual comments. Given that the inter-judge agreement is low (only 20% of judges comments were produced by more than one judge), the fact that TraumaTIQ produces 53% of the comments produced by at least one judge, and 70.3% of the comments produced by two or three judges suggests that its output is quite complete. Finally, I have demonstrated that TraumaTIQ generates its critiques in a timely manner.

Further evaluation of TraumaTIQ is needed to investigate the actual performance of the system in the emergency room. Some of the issues to be studied during this prospective evaluation will be discussed in the next chapter.

Chapter 9

Conclusions and Future Work

9.1 Contributions

In this thesis, 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.

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. The critiquing architecture I have proposed comprises a cycle of plan recognition, plan evaluation, and critique generation. These components are integrated with a goal-based reasoner and planner to provide them with information about what goals are *relevant* and how best to pursue them in the current situation.

9.1.1 Plan recognition

The plan recognizer uses knowledge about actions and goals in the domain, together with information from the reasoner about the specific situation, to infer a model of the user's goals and intentions from his proposed actions. The plan recognition algorithm makes the

assumption that the physicians are more likely to have appropriate goals but be addressing them in a sub-optimal way, than to be pursuing the wrong goals altogether. This assumption justifies the strategy of giving the physician the "benefit of the doubt" when their orders can be explained in terms of currently relevant goals.

By using the available evidence about what goals are relevant or almost relevant, and thus likely to be pursued, the plan recognizer is able to quickly develop a model of the plan to be evaluated. Other plan recognition systems that enumerate all possible plans to explain the observations require significantly more computation.

9.1.2 Plan evaluation

The plan evaluation component makes use of knowledge about the utilities and disutilities associated with domain actions and goals, together with knowledge of policy and practice guidelines and how they should shape behavior in a given situation, in order to identify errors that will then be mentioned in the critique.

The main contribution of the plan evaluator to the problem of decision support in general is the idea that the *potential significance* of an error to the outcome of the task should be considered in deciding whether to say something at all, and if so, how to say it. A further contribution is the analysis of the relationships between different types of error and the relevant evaluation functions and thresholds for comment. For example, it is more important to say something when an entire goal is being omitted, than when it is only being partially addressed, since in the latter case we can at least conclude that the physician is *aware* of the goal.

A different kind of contribution of the plan evaluator is as a tool for evaluating protocols for trauma management. Having developed a specification for evaluating errors, we can potentially use that specification to automatically validate management protocols in the future.

9.1.3 Critique generation

The critique generator converts the results of plan evaluation into a concise set of natural language critique comments. These comments include explanations regarding the

goals associated with actions being critiqued or recommended, the reasons for suggesting alternative procedures, and the reasons for recommending against certain actions.

It is very important for a decision support system functioning in a time-critical, complex situation to be able to communicate concisely and naturally with its users. For this reason, I have chosen to display the critiques in a series of short English sentences, which are intended to be authoritative yet polite. I am not making any claims regarding other aspects of these texts. In the next section I will discuss some future work which may be done in the area of language generation in TraumaTIQ.

9.1.4 TraumaTIQ

This architecture has been implemented in TraumaTIQ, a real-time critiquing system that can detect and respond to a range of planning failures in trauma management. 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.

An evaluation of TraumaTIQ shows that its comments correlate well with the comments made by human judges, when the judges also agree with each other. The evaluation also shows that the plan evaluation metrics are predictive of the ratings made by human judges on actual cases: TraumaTIQ produces more warnings in cases that were given a low rating by the judges.

9.2 Future Work

9.2.1 Extensions to the plan recognizer

The plan recognition system is currently able to ascribe goals to about 89% of actions in actual trauma management plans, 15% of which are actions that are not in TraumAID's plan at the time that they are ordered. Of the actions that are not explained, most of them are broad diagnostic tests such as X-rays, or surgical incisions which could lead to several therapeutic procedures. The goals underlying these actions could be understood at a more abstract level than is represented in TraumAID's knowledge base. Expanding

the representation of goals to include an abstraction hierarchy could thus improve the plan recognizer's ability to explain such actions when they are ordered.

I have also discussed in Chapter 4 the possibility of using probabilistic reasoning techniques to get a broader understanding of goals that are partially or almost relevant in other ways than the notion of potential relevance that I have used in TraumaTIQ.

9.2.2 Extensions to language generation

Explanation

The knowledge base developed for TraumAID 2.0 was designed for the purpose of generating quality decision support, not delivering it to physicians. As such, there are several areas in which additional knowledge could enhance the communicative abilities of TraumaTIQ.

To provide for better explanations, the knowledge base could be enhanced with more explicit information about the system's reasoning. This might include:

- Justifications for preferring for one procedure over another to address a particular goal. Is the preferred procedure faster, cheaper, more accurate, etc?
- Deeper explanations for contraindications and precedence constraints, so that TraumaTIQ can be clearer about why *not* to do an action.

Strategic generation

The focus of this dissertation has been on determining the *content* of the critiques, which is the first step in strategic generation. The sentences output by TraumaTIQ are sorted according to their significance and topic, but as I have discussed in Chapter 6, and have shown in the example in Chapter 7, in certain situations when these sentences are put together, the output can be somewhat redundant and lacking in coherence. Further work in the areas of discourse and sentence planning is needed to address these problems. For example, a discourse model would make it possible to select appropriate referring expressions for previously discussed entities to improve the overall coherence of the critiques over time. Sentence planning would allow for more concise presentation of information by combining related information into the same sentences.

Tactical generation

In TraumaTIQ, sentences are generated using templates whose slots are filled with translations of TraumAID concepts. These concepts are single units, which are often quite long and contain many sub-concepts. A more complex representation of these concepts would allow variations in sentence structure which are currently not possible. They would also make it possible to stress important sub-concepts in the output, either graphically or in spoken output.

9.3 Further retrospective evaluation

The comparison of TraumaTIQ's output with the comments of three expert judges that I presented in Chapter 8 indicates that TraumaTIQ does a fairly good job producing correct, clinically significant comments. However, it may be dangerous to be overconfident based on these results, because the judges' data I used were collected for a very different study, for which they were not asked to be complete in their notation of individual errors.

Additional evidence for this claim could be acquired by a more direct evaluation of the critiques produced by the system, such as the evaluation of the HyperCritic system that was done by van der Lei [90]. This evaluation was done in two stages: first have a group of judges and the critiquing system produce critiques for a set of cases, and then have the judges evaluate the critiques produced by the other judges and the system. The results of this study are an analysis of the quality of the critiquing system's critiques as compared to the critiques produced by human experts on the basis of the same information about the case.

Independently of an overall evaluation of the critiques, the plan recognition algorithm might also be evaluated by asking judges to give their opinion as to *what goal* various actions in a set of management plans were intended to achieve, and then comparing those judgments to the results of TraumaTIQ's plan recognizer.

9.3.1 Experimental evaluation

Computer scientists and anthropologists working in the area of Human-Computer Interaction (HCI) have pointed out the essential value of getting experience with a system in

the environment in which it is intended to be used as part of the development process [27, 26, 95]. In the near future, TraumaTIQ is due to be deployed experimentally in the Trauma Center at the Medical College of Pennsylvania. Experience with practicing trauma surgeons, using the system should provide a wealth of useful information for improving the system's performance and acceptability.

Mode of critique communication

The primary focus of this experiment will be the evaluation of alternative modes of communicating the critique to physicians. This will involve comparing the effectiveness, in terms of influencing physician's behavior, of displaying the critique to the scribe nurse who will then communicate it to the physician, versus using synthesized speech to communicate the critique directly to the physician. The two critiquing methods will also be compared to the direct presentation of TraumAID's plan to see if critiquing in general improves the system's effectiveness. There are both sociological and pragmatic issues influencing the choice of communicative mode.

On the sociological side, the scribe nurse has the advantage that she already has an accepted role in the trauma team, and has experience asking questions and making suggestions in such a way that the physicians are most likely to respond. On the other hand, the computer's *lack* of defined social role in the trauma team may prove to be an advantage in critiquing the physician's behavior. Since nurses traditionally have a subordinate role and are not qualified to offer medical advice, the physicians may not be willing to accept critiques from them, whereas the computer may be seen as more qualified by virtue of its being programmed by an expert.

On the pragmatic side, the scribe nurse has the advantage of being more directly aware than the computer of what is happening in the room. She can monitor the situation and make decisions about what critiques are valid based on information that may not be available to TraumAID. She can also more easily decide when and how to intervene with a critique. On the other hand, since the computer will be an additional member of the team, critiques coming directly from it via synthesized speech may be more salient to the physicians than comments coming from the scribe nurse.

Thresholds for evaluation

Another aspect of TraumaTIQ that may be adjusted as a result of experience with physicians is its "pickyness," or how many comments it produces. Physicians using the system will be asked to evaluate whether it comments too often, not often enough, or what specific items comments it made that they considered inappropriate. The plan evaluation module can then be adjusted if necessary. This could involve changing the thresholds for warning or commenting on errors in general, or changing the disutilities associated with individual goals. For example, if the physicians complain that TraumaTIQ is always telling them not to do an action X, when they don't consider it a very serious error to do X unnecessarily, it may be necessary to lower the cost of X in TraumAID's knowledge base.

It may also be necessary to adjust the timing of comments, particularly in the case of errors of omission. Currently TraumaTIQ waits no more than five minutes (depending on the urgency of the omitted goal) before commenting on an omission. Experience with physicians using the system should indicate whether this is an appropriate amount of time to wait or if the comment should be produced sooner or later.

Plan Recognition

The correctness of the inferences made by the plan recognizer regarding the physician's plans is an important issue that is difficult to assess with the currently available data, since there is no evidence for what goals the physicians were actually addressing when they executed particular actions. This question must therefore be addressed in future work during the field testing of TraumaTIQ. At that time, it will be possible to get information from physicians regarding what goals they were pursuing by debriefing them immediately following cases. This information will make possible a stronger test of the plan recognizer's correctness.

Interface issues

A number of aspects of the design of TraumaTIQ's output interface have necessarily been speculative up to this point. Experience with physicians using the system should help us determine the appropriate phrasing to use in the critique templates and the optimal

amount of time to persevere regarding a critique that the physician does not respond to.

9.3.2 User modelling

I have side-stepped 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, specific information about their expertise with various procedures, or their particular preferences for handling certain problems, or the types of errors they tend to commit. This information could be used to interpret the user's actions more accurately. For example, if the system knows that Dr. X prefers to address goal G using procedure P rather than Q, it can better understand why P is done, even if the system itself has a preference for doing Q to address G.

On a more general level, user modeling could be done to characterize the knowledge and behavior of whole classes of users, from medical students to experienced residents. This would allow TraumaTIQ to relax its assumption that the user has experience with trauma management, which is currently crucial to both the plan recognition and critique generation modules. A model of typical student reasoning, for example, would make it possible to recognize behavior that a more experienced physician would be less likely to exhibit, such as pursuing irrelevant or unjustified goals or using inappropriate procedures to address her goals. Additionally, a model of the user's knowledge could ultimately make it possible for the critique generator to produce explanations at different levels of detail for users with different expertise.

9.3.3 TraumaTIQ as a tutoring system

A final issue is the applicability of the critiquing architecture for use as a educational tool. Many people have worked on systems for tutoring skills in a particular task using simulations with automated feedback [19, 40, 48].

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. To this end, Professor Sandra Carberry has been developing a technique for automatically generating cases during a simulation so that the outcome is dependent on what the student does. In this way, it would be possible for the student to experience the negative effects of her errors more often than she would in real life (in the simulation, the failure to pursue a diagnosis can *cause* the diagnosis to be true, whereas in real life if the student fails to pursue a diagnosis, that diagnosis will not always be true.)

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 plan recognizer currently makes about the relevance 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.

9.4 Conclusion

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.

Bibliography

[1] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

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

[3] 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.

[4] G. Octo Barnett, James J. Cimino, Jon A. Hupp, and Edward P. Hoffer. DXplain: An evolving diagnosis decision-support system. *Journal of the American Medical Association*, 258:67–74, 1987.

[5] 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.

[6] Mathias Bauer. A Dempster-Shafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction*, 1995. to appear.

[7] 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.

[8] M. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

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

[10] Sandra Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the* 8^{th} *National Conference on Artificial Intelligence*, pages 471–478, Boston, MA, 1990.

[11] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[12] Eugene Charniak and Robert Goldman. Plan recognition in stories and in life. In *Proceedings Fifth Conference on Uncertainty in Artificial Intelligence*, pages 54–59, Windsor, Ontario, 1989.

[13] Eugene Charniak and Robert Goldman. A semantics for probabilistic quantifier-free first-order languages. In *Proceedings IJCAI-89*, pages 1074–1079, Detroit, MI, 1989.

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

[15] 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.

[16] W. J. Clancey and R. Letsinger. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 829–836, Vancouver, 1981.

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

[18] J. R. Clarke, R. Rymon, B. Webber, C. Hayward, T. Santora, D. Wagner, and A. Ruffin. The importance of planning in the provision of medical care. *Medical Decision Making*, 13(4):383, 1993. abstract.

[19] Chris Eliot and Beverly Park Woolf. Reasoning about the user within a simulation-based real-time training system. In *Proceedings of the Fourth International Conference on User Modeling*, Hyannis, MA, 1994.

[20] Rhonda Eller and Sandra Carberry. A meta-rule approach to flexible plan recognition in dialogue. *User Modeling and User-Adapted Interaction*, 2(1-2):27–53, 1991.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Göran Forslund. Designing for flexibility: a case study. *Expert Systems*, 12(1):27–37, February 1995.

[26] Diana Forsythe. Using ethnography to build a working system: rethinking basic design assumptions. In *Proceedings of the 15th Symposium on Computer Applications in Medical Care (SCAMC-91)*, pages 505–509, November 1991.

[27] Diana E. Forsythe and Bruce G. Buchanan. Broadening our approach to evaluating medical information systems. In *Proceedings of the 15th Symposium on Computer Applications in Medical Care (SCAMC-91)*, pages 8–12, November 1991.

[28] Jane M. Fraser and Philip J. Smith. A catalog of errors. *International Journal of Man-Machine Studies*, 37:265–307, 1992.

[29] Abigail S. Gertner. Real-time critiquing of integrated diagnosis/therapy plans. In *Proceedings AAAI Workshop on Expert Critiquing Systems*, 1993.

[30] Abigail S. Gertner. Ongoing critiquing during trauma management. In *Working Papers AAAI Spring Symposium on AI in Medicine*, 1994.

[31] Abigail S. Gertner. Responding to users' informational needs in time-critical situations. In *Proceedings of the Fourth International Conference on User Modeling*, 1994.

[32] Abigail S. Gertner, Bonnie L. Webber, and John R. Clarke. Upholding the maxim of relevance during patient-centered activities. In *Proceedings of the Fourth International conference on Applied Natural Language Processing*, pages 125–131, 1994.

[33] Abigail S. Gertner, Bonnie L. Webber, and John R. Clarke. Recognizing and evaluating plans with diagnostic actions. In *Working notes of the IJCAI-95 Workshop on Plan Recognition*, 1995.

[34] 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.

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

[36] Erik Hollnagel. The phenotype of erroneous actions: Implications for HCI design. In George R. S. Weir and James L. Alty, editors, *Human-Computer Interaction and Complex Systems*. Academic Press Ltd., London, 1991.

[37] Erik Hollnagel. The phenotype of erroneous actions. *International Journal of Man-Machine Studies*, 39:1–32, 1993.

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

[39] Patricia M. Jones, Rose W. Chu, and Christine M. Mitchell. A methodology for human-machine systems research: Knowledge engineering, modeling, and simulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(7):1025–1037, 1995.

[40] Randall W. Hill Jr. and W. Lewis Johnson. Situated plan attribution. *Journal of Artificial Intelligence in Education*, 1995. to appear.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] Michael J. Lincoln, Charles W. Turner, Peter J. Haug, Homer R. Warner, John W. Williamson, Omar Bouhaddou, Sylvia G. Jessen, Dean Sorenson, Robert C. Cundick, and Morgan Grant. Iliad training enhances medical students' diagnostic skills. *Journal of Medical Systems*, 15(1):93–110, 1991.

[47] 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.

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

[49] 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.

[50] Harold C. Lyon, James C. Healy, James R. Bell, Joseph F. O'Donnell, Edward K. Schultz, Robert S. Wigton, Frank Hirai, and J. Robert Beck. Findings from an evaluation of PlanAlyzer's double cross-over trials of computer-based, self-paced, case-based programs in anemia and chest pain diagnosis. In *Proceedings of the 15th Symposium on Computer Applications in Medical Care (SCAMC-91)*, pages 88–93, November 1991.

[51] 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.

[52] 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.

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

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

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

[56] Gianpaolo Molino, Marco Ballare, Patrizia E. Aurucci, and Vittorio Ripa Di Meana. Application of artificial intelligences techniques to a well defined clinical problem: jaundice diagnosis. *International Journal of Biomedical Computing*, 12:189–202, 1990.

[57] 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.

[58] R. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Common Sense World*, pages 319–358. Ablex Publ., Norwood NJ, 1985.

[59] L. Morgenstern. Knowledge preconditions for actions and plans. In *Proc. 10th Int'l. Joint Conf. on Artificial Intelligence*, pages 867–874, Milan Italy, 1987.

[60] 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.

[61] 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.

[62] 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.

[63] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, Calif, 1988.

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

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

[66] H. Pople. Heuristic methods for imposing structure on ill-structured problems. In P. Szolovits, editor, *Artificial Intelligence in Medicine*, pages 119–190. Westview Press, Boulder, CO, 1982.

[67] 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.

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

[69] Lance A. Ramshaw. A three-level model for plan exploration. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 39–46, Berkeley CA, June 1991.

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

[71] J. T. Reason. *Human Error*. Cambridge University Press, Cambridge, 1990.

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

[73] William B. Rouse, Norman D. Geddes, and Renwick E. Curry. An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction*, 3:87–122, 1987-1988.

[74] R. Rymon, B. Webber, and J. R. Clarke. Progressive horizon planning. In *Proceedings of the 7th Israeli Conference on Artificial Intelligence and Computer Vision*, Tel Aviv, Israel, 1990.

[75] Ron Rymon. *Diagnostic Reasoning and Planning in Exploratory-Corrective Domains*. PhD thesis, Department of Computer & Information Science, University of Pennsylvania, 1993. Appears as Technical Report MS-CIS-93-84.

[76] Glen Shafer and Judea Pearl, editors. *Readings in Uncertain Reasoning*. Morgan Kaufman, San Mateo, Calif, 1990.

[77] Yuval Shahar and Mark A. Musen. Plan recognition and revision in support of guideline-based care. In *Working notes of the AAAI Spring Symposium on Representing Mental States and Mechanisms*, pages 118–126, Stanford, CA, 1995.

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

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

[80] Barry G. Silverman. *Critiquing Human Error*. Academic Press Ltd., London, England, 1992.

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

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

[83] Sampath Srinivas and Jack Breese. IDEAL: Influence diagram evaluation and analysis in Lisp: Documentation and users guide. Technical Report 23, Rockwell International Palo Alto Laboratory, 1992.

[84] Lucy A. Suchman. *Plans and situated actions: the problem of human-machine communication.* Cambridge University Press, Cambridge, 1987.

[85] William Swartout. XPLAIN: A system for creating and explaining expert consulting programs. Research Report, Information Sciences Institute, Marina del Rey, California, 1983.

[86] 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.

[87] 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.

[88] Peter van Beek. A model for generating better explanations. In *Proceedings ACL-87*, 1987.

[89] Peter van Beek, Robin Cohen, and Ken Schmidt. From plan critiquing to clarification dialogue for cooperative response generation. *Computational Intelligence*, 9(2):132–154, 1993.

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

[91] Vijay Vasandani and T. Govindaraj. Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(7):1077–1095, 1995.

[92] 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(2):145–163, April 1992.

[93] Robert Wilensky. *Planning and Understanding: A computational approach to human reasoning.* Addison-Wesley, Reading, MA, 1983.

[94] Robert Wilensky, David Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The Berkeley UNIX Consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[95] Jeremy Wyatt and David Spiegelhalter. Field trials of medical decision-aids: potential problems and solutions. In *Proceedings of the 15th Symposium on Computer Applications in Medical Care (SCAMC-91)*, pages 3–7, November 1991.