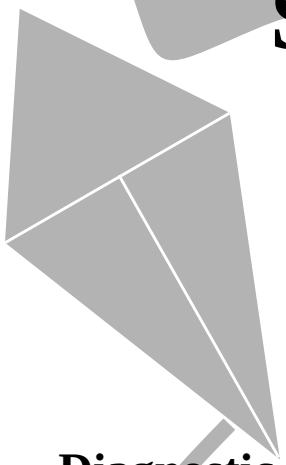
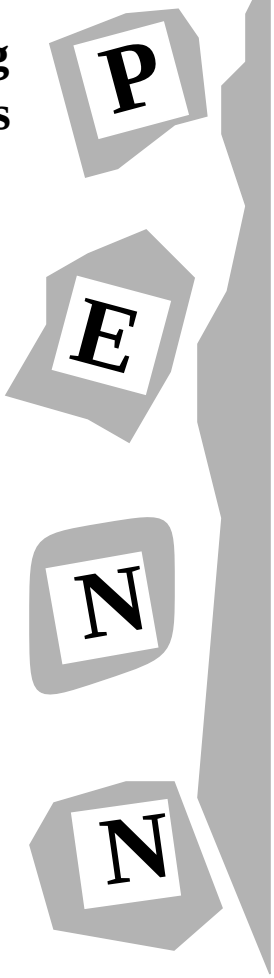
The Institute For Research In Cognitive Science



Diagnostic Reasoning and Planning in Exploratory-Corrective Domains (Ph.D. Dissertation)



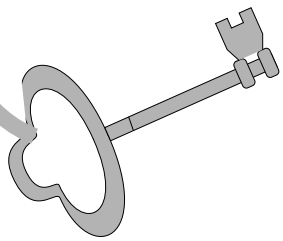
by

Ron Rymon

University of Pennsylvania
Philadelphia, PA 19104-6228

November 1993

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



University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 93-40

DIAGNOSTIC REASONING AND PLANNING IN EXPLORATORY-CORRECTIVE DOMAINS

Ron Rymon

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

1993

Bonnie L. Webber

Supervisor of Dissertation

Mark Steedman

Graduate Group Chairperson

To Tali and My Parents

Acknowledgements

First and foremost, I am very grateful to my advisor, Professor Bonnie L. Webber, and to Professor John R. Clarke MD, for their invaluable guidance and support throughout the development of this work and indeed throughout my own development as a scientist thus far. I owe much of the content of this document to many hours of discussions with Bonnie and John. In the past two years, Bonnie has effectively balanced between pushing me to finish this work and being patient and supportive of my new research interests. It has been my pleasure working with them both.

I am also grateful to my committee members: Professors Peter Szolovits, John Clarke, Gregory Provan, and Mitchell Marcus. The direction I have taken in this work, most notably my recent investigation of abstract diagnosis-and-repair strategies, was greatly influenced by their many comments and suggestions.

I wish to thank the faculty, staff and students in the Computer and Information Science Department and in the Institute for Research in Cognitive Science at Penn for their teachings, support, colleagueship and friendship during my years there. Special thanks are due to members of the TraumAID group throughout the years: Professors Webber, Clarke and Provan, Mike Niv, Abigail Gertner, Jonathan Kaye, Stefannie Neumann, Charlie Ortiz, Mila Ibanez, Jim Gamble, Jim Markeley, Kaiti Moore, Adam Bell, Alon Luss, Ernie Sparks, Rich Tuttle and Jim Farrelly amongst others.

In specific Chapters, I have benefited from the following help. The formalization

of GDD (Chapter 3) has greatly benefited from Matt Ginsberg's work on MVL; I thank Matt for helping me to understand MVL, and for a number of discussions on its application to GDD. The work on GDD has also benefited from discussions with participants in the annual Diagnosis Workshop. The experimental evaluation of TraumAID 2.0 (Section 6.2) was designed and carried out by John Clarke and his staff at the Medical College of Pennsylvania. Doctors Catherine Hayward, David Wagner, and Thomas Santora served as judges. The more recent work on abstract strategies (Chapter 7) was suggested by Peter Szolovits. In this work, I have also benefited from numerous replies to my inquiry in the `ai-medicine` list.

On the personal side, I am most indebted to Tali – my wife and best friend – who came with me to the U.S. wholeheartedly, despite the many unknowns about her own future. We ended up having lots of great times throughout the past five years, and Tali will soon complete her own Ph.D., in Marketing, at The Wharton School of the University of Pennsylvania. The hard times along the way were much easier with Tali at my side, providing me with all the support she could.

I am also greatly indebted to my parents who planted the seeds of curiosity in me and who then had to put much of their energy into its satisfaction. I wish to express my deep love to my parents, sisters, members of my greater family and Tali's family; I thank them all for their love and support. I wish to especially thank Esther Eibszyc for her encouragement and support. I am forever grateful to Bernice and George Beckerman for helping me say "yes" to the challenge and for subsequently easing so much the first steps in its fulfillment.

Parts of this work were supported by an Army graduate fellowship, ARO Grant DAAL03-89-C0031PRI.

Ron Rymon

September 1993

Abstract

DIAGNOSTIC REASONING AND PLANNING IN
EXPLORATORY-CORRECTIVE DOMAINS

Ron Rymon

Bonnie L. Webber (Supervisor)

I have developed a methodology for knowledge representation and reasoning for agents working in exploratory-corrective domains. Working within the field of Artificial Intelligence in Medicine, I used the specific problem of diagnosis-and-repair in multiple trauma management as both motivation and testbed for my work.

A reasoning architecture is proposed in which specialized diagnostic reasoning and planning components are integrated in a cycle of reasoning and action/perception:

1. A *Goal-Directed Diagnostic* (GDD) reasoner which is predicated on the view that diagnosis is only worthwhile to the extent that it can affect repair decisions and that goals can be used to focus on such. Rather than focusing on a diagnosis object as the primary purpose of the diagnostic process, the GDD reasoner is tasked primarily with generating goals for the planner and with reasoning about whether these goals have been satisfied.

2. A *Progressive Horizon Planner* (PHP) which works by constructing intermediate plans via a combination of plan sketching and selection/optimization sub-processes, and then adapting these plans to reflect new information and

goals. For the plan sketching sub-part, I propose a selection-and-ordering planning/scheduling paradigm, taking advantage of the limited interaction between goals.

I have implemented this architecture and reasoning components in TraumAID 2.0 – a consultation system for the trauma management domain. In a blinded comparison, out of 97 real trauma cases, three trauma surgeons have judged management plans proposed by TraumAID 2.0 preferable to the actual care by a ratio of 64:17 and to plans generated by its predecessor TraumAID 1.0 by a ratio of 62:9.

Contents

List of Figures

Chapter 1

Introduction

1.1 Reasoning and Acting in Exploratory-Corrective Domains

This dissertation represents an attempt to address the general problem of reasoning and acting in exploratory-corrective domains by studying the requirements of a specific such domain – the management of multiple trauma patients. The multiple trauma management domain both motivated my work and served as a testbed for new ideas.

By exploratory-corrective domains I refer to domains in which an agent should intersperse exploratory activity with activity aimed directly at achieving the agent's goals. In a way, almost every domain in which we would expect artificial agents to operate is exploratory-corrective. Exploration is necessary because it is unrealistic to assume that an agent be equipped with a complete description of the environment in which it will operate and with a precise model of the way in which the environment will change while it operates there. Further, in many domains, it is impossible to completely pre-specify an agent's goals. These goals, or at the very least their operational specification, may have to be determined through exploration. In many

domains agents must adopt, abandon, or re-prioritize their goals while they act, in response to new situations or newly acquired information. What makes exploratory-corrective domains hard is that, in committing its limited resources, an agent has to continuously choose between further exploration and characterization on one hand and direct pursuit of its corrective goals on the other.

Certain *diagnosis-and-repair* domains, ones in which an agent is charged with restoring the functionality of a broken system or treating an ill patient, are excellent exemplars of such domains. The restoration goal in such domains is often far from being well-specified. In medicine, for example, this goal may be phrased as abstractly as: *"make the patient healthy again"*, giving little or no clue about the specifics of the problem and what would constitute an acceptable solution. In addition to goal characterization, exploration may also be necessary to determine other conditions of the system and characteristics of the environment that might interact with the problem or with repair strategies. Of course, in some diagnosis-and-repair domains, complete specification *is* possible, and ramifications of uncertainty and unpredictability can be ignored. There probably are also domains where one can afford complete characterization *prior* to repair. In this work, though, I concentrate on domains where there is an inherent tradeoff between exploration and repair and in which there is a rich interplay between the two.

The remainder of this introduction chapter is structured as follows: Section 1.2 overviews the multiple trauma management problem domain by using a fairly elaborate case-example in which its exploratory-corrective features are clearly demonstrated; Section 1.3 overviews the main claims and contributions of this thesis; and Section 1.4 overviews the contents of the remainder of this document.

1.2 The Problem

1.2.1 The Need for Aid in Trauma Management

As of 1985, injuries, accidental and intentional, are the third largest cause of death in the United States and result in more years of human life lost in the U.S. than any other disease [3, 60]. Hitting young people more than all other diseases combined, the economic loss reached $130 billion annually, in the U.S. alone.

Trunkey [153] shows a trimodal distribution of injury-related deaths:

1. immediate, as a result of the injury;
2. within the first few hours of injury; and
3. as a result of later complications.

Since immediate deaths are only amenable to injury prevention, and since part of the complications result from less than adequate earlier care, it is recognized that the first few hours of injury are critical to the patients' survival. Indeed, West *et al.* [158] found that 30-40% of trauma deaths are preventable by the delivery of rapid expert care.

Resuscitative care begins in the site of injury and continues into the hospital's emergency room, where a patient would typically spend the first "*golden*" hour in initial assessment and resuscitation. The American College of Surgeons' *Advanced Trauma Life Support* (ATLS) course [2] addresses the first peak of injury-related deaths. It is used to train care providers in the initial evaluation, resuscitation, and stabilization of severely injured patients. The Guardian project [68] is aimed at monitoring patients in the intensive care unit and is thus addressing (partially) deaths associated with the third peak. The goal of our TraumAID project [155] is to help reduce deaths and serious morbidity in the second phase, often referred to as the *initial definitive management* phase.

During the initial definitive management phase, in the hospital, a patient undergoes a more complete series of tests and treatments designed to diagnose and treat the particular combination of problems sustained. It ends when the patient is transferred to the operating room for major surgery, or to the trauma unit for continued care, or when the patient is discharged.

Computerized decision support can help surgeons manage patients with major trauma in at least three ways:

1. It can provide expertise that is otherwise lacking – for example, in rural areas, or outside normal working hours, where a trauma patient may be admitted by a resident and/or a physician whose main expertise does not involve trauma.

2. It can be used as an on-line *Quality Assurance* (Q.A.) device, monitoring deviations from the required standard of care. Real-time Q.A. is of particular importance in trauma management since residents' lack of expertise and experience is often augmented by fatigue. After the fact, it can serve as a useful educational facility.

3. It can be used to *standardize* care [148]. Standardization, if done properly, can reduce health-care costs[1].

The potential for decision support system is highlighted by the preliminary empirical evaluation of TraumAID 2.0 (cf. Section 6.2) which shows that surgeons prefer its management plans to those followed in actual patient care. To date however, with the exception of TraumAID 1.0, most decision support systems for the initial definitive management phase have come in the form of scoring algorithms (e.g. [12, 107]). The most widely used aide is the *Trauma Score* developed by Champion and Sacco [12], which itself incorporates the *Glasgow Coma Score* [151], a scoring system for head injuries. The Trauma Score system has also been used in scoring systems developed

[1]One of the conclusions of Clarke *et al.* [17], for example, is that surgical residents request more diagnostic tests than really needed.

for *post-hoc* evaluation and quality assurance purposes [9, 13]. However, given the large number of possible diagnoses[2], and given that a patient may suffer from a *number* of injuries or a number of problems caused by the same injury, these scoring devices can hardly be expected to cover all possible combinations. A knowledge-based approach may have the potential to provide a more flexible solution.

1.2.2 Exploratory-Corrective Trauma Management

Consider the following (fictitious) case-example which, although not characteristic in its complexity, is useful for demonstrating many of the key exploratory-corrective features of the multiple trauma management domain and which I will revisit several times throughout this dissertation:

> JR, a male in his early forties, was brought to the Emergency Room in an unstable condition (*shock*[3]) after being shot twice in the chest. He had decreased breath sounds on both sides, and his neck veins were distended. JR also suffered from a broken arm, apparently twisted when he fell to the pavement in front of his office building.

The above text reveals two things: (1) that when a patient arrives in an Emergency Room, information available about the patient and his/her injuries is often incomplete and cannot support any deep diagnosis – thus the need for exploration; and (2) that trauma management often involves multiple injuries – thus the need for planning to resolve possible conflicts between competing diagnostic and/or therapeutic needs.

> Initial assessment showed muffled heart sounds, however normal pulses and normal abdominal findings. The attending physician had hypothesized that the shock resulted from a *tension pneumothorax*[4] in one or both sides of the chest and/or from a pericardiac injury[5]. Secondary

[2] There are over 1000 different diagnoses [1].

[3] By shock I generally refer to low blood pressure.

[4] A pneumothorax is a condition in which air leaks into the chest cavity between the chest wall and the lung to the point beyond which the return of blood to the heart is affected by the resulting increase in pressure.

[5] A *pericardial tamponade* is a condition in which blood accumulates in the pericardial sac, interfering with the heart's function.

> concerns were that a bullet may have injured organs internal to the chest and/or abdomen.

This latter part of JR's management exemplifies a *routine* physical examination, as opposed to a more elaborate planning of costly and risky procedures. Also note how a set of potential diagnoses is identified – a common diagnostic practice, often referred to as *hypothesis generation*[6].

> The most urgent need at this point was clearly the re-stabilization of JR. Since a tension pneumothorax was determined to be the most likely cause of shock, a needle aspiration of both sides of the chest was urgently called for. (If that hypothesis were confirmed, a chest tube would be inserted to further ease the pneumothorax.) A needle aspiration of the pericardial sac was contingently planned, in case the shock was not relieved. Finally, X-ray studies of the chest and abdomen were planned to rule out other injuries. The arm injury was ignored at this point.

Several additional features of the multiple trauma management domain are illustrated by the above segment:

First, that *active* exploration is often necessary. Importantly, diagnostic *activity* may also affect the patient's state, not only our knowledge about it. In some cases, diagnostic activity may have an adverse effect, either directly or due to the time and resources it consumes. In other cases, a diagnostic action may also have a certain *therapeutic* effect; a needle aspiration of the chest, for instance, may provide partial care in patients who *are* suffering from a tension pneumothorax.

Second, with respect to the *choice of diagnostic means*, hypotheses may sometimes be testable in several alternative ways. For example, a tension pneumothorax can also be detected on an X-ray film. Conversely, a single test can often provide information on more than one condition. For example, a single X-ray can often shed light on several hypotheses. Thus, in selecting a test, there may be an efficiency to be gained by selecting tests that cover more than a single condition. Other important factors

[6] See Section 7.2.

in test selection are accuracy, expediency, the risk and pain involved, dollar cost, etc. For example, in JR's case, an invasive needle aspiration procedure was chosen over a less painful, but not as expedient, X-ray study.

Third, considering further *interactions of competing needs*, some injuries are more urgent and/or more important than others. We have just seen that urgency can affect the *choice* of means (preferring more expedient procedures). Urgency and importance, however, can also affect the *order* in which various needs are pursued. In the multiple trauma management domain, it is common to categorize problems as being related to either airway, or breathing, or circulation, etc. (the so-called ABC's of trauma management), and to attend to them in that order of importance. Thus on one hand the immediate pursuit of the pneumothorax (urgent as a potential cause of shock and important by virtue of being an airway-related problem), and on the other hand the neglect of the broken arm. Also notice that diagnostic tests aimed at competing hypotheses, e.g. pneumothorax and pericardial tamponade as alternative causes of shock, are sometimes ordered by their assessed *likelihood*.

Continuing with JR's management:

> The needle aspiration of the chest came up negative for the right, but positive for the left chest. Nevertheless, despite the chest decompression, JR remained in a state of shock. The diagnostic efforts were thus shifted to a pericardial tamponade as an alternative cause of shock. In the meanwhile, a chest tube was inserted into the left chest to prevent deterioration of the pneumothorax condition. In addition to aspirating the pericardial sac, the current management plan also called for active monitoring of the chest tube (visually and via an X-ray, to see that it was correctly placed and functioning) and of the pneumothorax condition, as well as for the remaining X-ray studies.
>
> The needle aspiration of the pericardial sac proved to be effective in restabilizing JR, implicating pericardial tamponade as the true cause of shock. JR's chest was now continuously decompressed, to allow further diagnosis until a heart surgery can be performed.

In the above segment, we see a shift of attention from a *positive* pneumothorax to a *potential* pericardial tamponade. It is an excellent example of a diagnostic action

(aspiration of the chest) that "succeeded" but yet does not address completely the purpose for which it was taken. This part of the management also demonstrates occasional need to *intersperse* diagnosis and treatment. Intentionally incomplete treatment is also demonstrated by the temporary decompression of the pericardial sac. The definitive heart surgery is postponed to allow further diagnosis and treatment of other injuries. This delay is also motivated by other planning principles: first, logistically, such operation can only be done in the operating room; and second, as will soon be demonstrated, future diagnoses may require operations that could be more efficiently combined with this surgery.

> While the chest tube achieved its objective of relieving the pneumothorax, the massive flow of blood from the chest indicated a *massive hemothorax*[7]. A left *thoracotomy*[8] was then planned to treat this condition. However, given the concurrent need for a heart surgery, both operations were slightly altered to include a single incision across the chest such that both the heart and the left chest cavity are exposed.

We have seen before a diagnostic action with therapeutic effects. What we see here is an example of *empirical therapy*: the use of treatment as a diagnostic device by observing the patient's response. The chest tube, which was inserted for mainly therapeutic reasons, provided important diagnostic information. We also see efficiency gained by combining treatment for two problems.

> A chest X-ray, performed next, showed a hemothorax condition developing in the right chest, although not as massive as the one on the left. It further showed one bullet, in the chest midline. The other bullet did not appear on that film. Two more X-rays were then taken: a lateral chest X-ray to establish the exact location of the first bullet near the spine[9]; and an abdominal X-ray to determine if the second bullet lodged in the abdominal cavity. At this point, the current management plan called subsequently for a urinlaysis, and then for a chest surgery and a *laparotomy*[10] to repair the diaphragm and to explore other possible injuries in the abdomen.

[7] A hemothorax is internal bleeding in the chest cavity which results in blood accumulation between the lungs and the chest, in turn collapsing the lung.

[8] A thoracotomy is a chest operation.

[9] The 3D location can only be determined via the combination of both X-rays.

[10] A laparotomy is abdominal surgery.

An important principle demonstrated here is that of *limited* diagnosis. Notice that from the information available so far, the entry wounds and the bullet locations cannot be correlated with certainty. In particular, we are uncertain as to which of the two bullets has made its way to the heart and which headed down to the abdomen. As a result, we cannot determine which of the organs internal to the abdomen were injured; in particular, whether the left or right part of the diaphragm needs to be repaired. However, since such determination would *not* have affected the management plan (it would not change the type of surgery procedure chosen – a laparotomy), it is not pursued further.

A related phenomenon, that of *parsimonious repair*, is also demonstrated here. A chest tube, normally inserted to treat a hemothorax, is excluded given that the patient is anyway headed for a surgery that will expose the right chest.

> The urinalysis came out negative, and JR finally underwent a bilateral transverse sternotomy and a laparotomy. He was then transferred to the Intensive Care Unit for monitoring.

To summarize, this case is characteristic of exploratory-corrective domains in the kind of issues involved in reasoning and acting. It starts with incomplete information and ill-specified corrective goals. To decide which goals to pursue, and how to pursue them, an intelligent agent will typically have to start off with some exploration. After reaching a certain level of knowledge, such an agent will begin pursuing repair goals, while continuing exploration as required. As I will argue, what is probably the most important characteristic of intelligent exploratory-corrective behavior is that the decision *whether* to pursue a diagnostic test or a repair procedure, *when* to pursue it, to which *extent*, and using *what means*, is ultimately based on its potential effect on repair objectives. Indeed, this is a most important underlying principle in TraumAID 2.0, a new consultation system for the trauma domain. Next, after briefly describing its predecessor, TraumAID 1.0, I formalize these principles in a general framework for diagnostic reasoning and planning.

1.2.3 TraumAID 1.0: A Rule-Based Consultation System

Developed between the years 1984 and 1988, TraumAID 1.0 is a *rule-based* consultation system for multiple trauma management. Its scope consists of the initial definitive management of gunshot and stab wounds to the chest and abdomen. In a first feasibility study, management generated from a set of 122 rules for abdominal trauma management was judged to perform better than chief surgical residents [18].

In many ways, TraumAID 1.0 was modeled after some of the leading medical expert systems of the time. Like MYCIN [140], the pioneer expert system for diagnosis and treatment of infectious disease[11], TraumAID 1.0 used production rules. It had two types of rules: *suspect rules* were used in a forward-chaining manner to link evidence to suspicions whereas *conclude rules* were used in both forward- and backward-chaining to prove, or rule out, suspicions. When used in a backward-chaining fashion, conclude rules served, much like in *prolog*, to control information gathering. Simplified examples of suspect and conclude rules are shown below:

1. Suspect pericardial tamponade in patients that suffer a chest wound and have muffled heart sounds:

   5405? *Pericardial_Tamponade :-*
   Chest_Wound,
   Muffled_Heart_Sounds.

2. Conclude a pericardial tamponade if, in addition, signs of pericardial tamponade are observed in an ultrasound examination and if the patient is *not* (— sign) in shock:

[11] Interestingly, Shortliffe and Rennels say of MYCIN in the *Encyclopedia of Artificial Intelligence*, S. Shapiro ed., p. 589: "Although MYCIN is often described as a diagnostic program, its principal motivation was therapy planning".

5415: *Pericardial_Tamponade :-*

Chest_Wound,

Muffled_Heart_Sounds,

Ultrasound_Effusion(Positive),

— *Shock.*

TraumAID 1.0's inference and information gathering paradigms also resembled those of INTERNIST-I [103]. INTERNIST was a major research effort in automating reasoning in internal medicine. It used tables describing causal and associational relationships between various diseases and their symptoms. INTERNIST distinguished four different inference paradigms: conclude, pursue, rule-out, and discriminate. For evidence-gathering, INTERNIST coarsely classified information acquisition into four categories: obtainable from patient's history, requiring physical examination, non-invasive lab procedures and invasive procedures. This categorization was useful because the kind of considerations involved in choosing to perform a physical examination are clearly different from those involved in picking an invasive procedure. For its part, TraumAID 1.0 made use of a static ranking of all information-acquisition actions to determine their respective ordering. However, since under some circumstances (combinations of injuries) this order must be altered, TraumAID 1.0 had to resort to elaborate rule-blocking and filtering maneuvers. For example, in the above conclude rule (5415), the no-shock provision has nothing to do with the predicating the presence of a pericardial tamponade and is only used so as to avoid taking the time for an ultrasound when a patient is unstable.

During TraumAID 1.0's development, especially when its scope was extended from strictly abdominal injuries to injuries of both abdomen and chest, the awkwardness of elaborate rule blocking and filtering became unacceptable. It was recognized that rules are inadequate as a single representation and reasoning vehicle. TraumAID 2.0's main contribution over TraumAID 1.0 is in the area of representation and reasoning

for controlling actions. Instead of complicating rules with non-declarative control clauses, as above, TraumAID 2.0 uses planning principles to coordinate activity for a given combination of diagnostic and therapeutic needs.

It is important to note that this control problem is not unique to TraumAID 1.0. In a recent article [30], Davis, Buchanan and Shortliffe note the following:

> *Accommodating both procedural and inferential knowledge.*
>
> Some of the things we know appear to be fundamentally inferential (e.g. if the patient is a college student complaining of fever and fatigue, the disease if likely to be mononucleosis), while others have a strongly procedural character ("to change the oil in your car, first get a large disposable container, then ..."). Each of these of course has a natural representation; the difficulty arises when tasks have both sorts of knowledge. In MYCIN, for example, the diagnostic task is fundamentally inferential, but there is also an overall seven-step procedure that structures the diagnosis [24]. The difficulty of dealing with both forms of knowledge at once is displayed by MYCIN's relatively obscure solution to the problem: it produces the seven steps as a side-effect of the backward chaining process and several carefully constructed rules.
>
> One simple reason for this implicit encoding is that rules offer no easy way to express ordering information. A deeper reason is the dearth of fundamental insight about representations that facilitate expressing both forms of knowledge and allow them to work together intimately. The issue is not simply one of providing the computational machinery to allow rules to call procedures, or vice versa, but the lack of representational machinery that can guide our understanding, expression and use of that knowledge. Integration at the level of computational machinery is easy; the needed synergy is at the knowledge level.

In this dissertation I hope to have contributed, at the knowledge level, to ways in which to represent and use knowledge in domains which require both inferential and procedural reasoning. The Exploratory-Corrective Management (ECM) reasoning architecture and the accompanying diagnostic reasoning and planning frameworks (Chapters 2-4) provide a computational machinery that I believe can be used to "intimately" integrate reasoning of both types. In Chapter 5, I outline some knowledge-level guidance as to how to decompose the domain knowledge (both representation- and reasoning-wise) to effectively use this framework.

1.3 Claims and Contributions

As exemplified by JR's case (Section 1.2), artificial agents working in exploratory-corrective domains will have to continuously reason to *characterize* their environment, and to formulate their objectives and intentions. They will also have to continuously decide which *actions* to take. My central claim in this dissertation is that Artificial Intelligence diagnostic reasoning and planning capabilities can be integrated effectively in a reasoning architecture for exploratory-corrective agents. In Chapter 2, I first show that to be effective such agents must continuously cycle between diagnostic reasoning, planning, and action. I then present an architecture for such agents in which the overall reasoning task is decomposed into that which is diagnostic in nature and that which is concerned with the choice and ordering of action (planning), with goals serving as a natural interface between the two. In Chapters 3-4, I fit this architecture with specific diagnostic reasoning and planning paradigms which I developed to address the particular needs of exploratory-corrective domains. In Chapter 5, I discuss a methodology for effectively encoding exploratory-corrective behavior in this architecture.

To support this claim as a whole, I have implemented TraumAID 2.0 – a consultation system for the trauma management domain which employs an *Exploratory-Corrective Management* (ECM) architecture. During its construction, TraumAID 2.0 was validated on 270 hand-crafted cases covering many combinations of penetrating injuries to the chest and abdomen. Then, in a later study described in Chapter 6, management plans proposed by TraumAID 2.0 for real trauma cases were compared to the actual care as well as to plans proposed by TraumAID 1.0. A panel of three trauma surgeons, not otherwise associated with the project, judged TraumAID 2.0's plans to be significantly superior to both the actual care and TraumAID 1.0's plans.

With respect to a particular form of diagnostic reasoning suitable for diagnosis-and-repair domains, I have pointed out in Section 1.2 that in managing trauma patients, diagnosis must only be pursued so long as it can affect repair decisions.

This principle, which becomes even more important when considering diagnostic *activity*, can be extended to repair decisions: repair is only worthwhile to the extent that it can affect the final outcome. In Chapter 7, I identify similar patterns of behavior in several abstract diagnosis-and-repair strategies.

I claim that, in diagnosis-and-repair domains, the more important role of reasoning is to recommend appropriate *actions*. I therefore claim that it is more beneficial for a diagnostic reasoner to focus on *goals* than on a characterization of the patient's condition. I also claim that, in addition to their role as a natural interface between a diagnostic reasoner and a planner, goals can also facilitate *focusing* diagnostic and therapeutic activity. In Chapter 3, I show that reasoning for goal *specification* and reasoning about goal *satisfaction* are natural characterization tasks if one distinguishes one's belief in whether a statement holds from one's attitude towards acquiring information about it, or toward achieving a state in which it holds. I present a logical calculus which cleanly separates belief from attitude. I then use this calculus to formalize *Goal-Directed Diagnosis* (GDD), a formulation of diagnosis in which goals are explicitly targeted by the diagnostic engine. There, and later in Chapter 7, I show that GDD can be used to focus activity on goals that are both relevant and potentially contributing. An early version of GDD is implemented in TraumAID 2.0. The particular formulation presented in Chapter 3 has been implemented and tested separately.

With respect to the interaction between the diagnostic reasoner and the planner, I claim that explicit representation and reasoning about goals can sometimes facilitate a reduction in the number of goals that need to be planned for. In particular, in Chapters 3, 5, and 7, I present examples where the interaction between goals can be resolved in the goal-level, without forcing the planner to consider alternative means for addressing each goal. In TraumAID 2.0, I represent a goal *hierarchy* which allows inhibiting some goals at the presence of others. Chapter 5 discusses goal-level resolution in further detail.

With respect to particular forms of planning which are suitable for exploratory-corrective domains, I claim that the planning process should be *partial-global*. Planning must be partial since some relevant goals may yet have to be determined. In addition, given the current computational complexity of planning, a reasonable planner will likely also be partial in its search for a plan. Nonetheless, in domains such as trauma management, a planner must also be global in that its plans must always address *all known* goals. On one hand, complete plans provide a constructive way to evaluate the relative value of alternatives to individual steps when combined with steps aimed at other needs. On the other hand, spelling out a complete plan is also essential for attaining a reasonable level of communication with the interacting physician. In Chapter 4, I propose an incremental planning framework in which intermediate plans are constructed in each cycle of the ECM architecture, followed partially, and then adapted to reflect new information and changes in goals. In constructing intermediate plans, this *Progressive Horizon Planning* (PHP) framework first computes a plan *sketch*, and then invests its computational resources in improving the plan's initial segment. The reasoning behind this approach is that not only are they next to be executed, these first few actions in the plan are also least likely to be affected by uncertainty and/or unpredictability. I have implemented the PHP framework in TraumAID 2.0.

For the purpose of the plan sketching process, I claim that it is useful to represent goals so that their interaction is limited to competition for resources, and compatibility considerations. In Chapter 4, I show that under these assumptions, alternative sub-plans (procedures) and sub-strategies (conditionalized progressions of such) can be pre-constructed and prioritized for each goal individually. In that case, the planning problem turns out to be that of *mediating* between competing diagnostic and therapeutic needs. That is, the choice of a combination of procedures for a given combination of goals and the ordering of these procedures in a plan. In Chapter 4, I show that the choice and ordering sub-tasks can be formulated separately and then operationally interleaved in a simple *selection-and-ordering* planning algorithm. Chapter 5

expands on a representational methodology for exploratory-corrective behavior in the ECM architecture that is accommodated by the limited interaction assumption. The selection-and-ordering algorithm is also implemented in TraumAID 2.0's planner.

Finally, in Chapter 7, I show that important diagnostic strategies, and strategies that mix diagnosis and repair can be implemented in a goal-directed way. I show that goals play a very important role in these strategies and that it is thus only natural to encode them in GDD. This study has also resulted in an intriguing collection of diagnosis-and-repair strategies, which I expect to be useful to test other hypotheses and/or methodologies developed for assembling or critiquing action plans in exploratory-corrective domains.

1.4 An Overview of the Remainder

Chapter 2 presents the Exploratory-Corrective Management (ECM) reasoning architecture. Chapter 3 presents a formulation of the Goal-Directed Diagnosis framework which is based on Ginsberg's Multi-valued logics. Chapter 4 formulates the planning framework used in TraumAID 2.0: Section 4.1 describes the Progressive Horizon Planning technique, and Section 4.2 describes the Selection-and-Ordering plan sketching framework. Then, in Chapter 5, I describe a methodology in which these two frameworks can be *used*: on-line mediation of explicitly encoded individual strategies.

Chapter 6 first exemplifies TraumAID 2.0 using JR's case, and then presents empirical results from a comparative study in which TraumAID 2.0's plans were compared to the actual care provided in trauma cases, as well as to those of its predecessor TraumAID 1.0.

Chapter 7 presents a catalogue of diagnosis-and-repair strategies and discusses their implementation in the ECM framework.

Chapter 8 concludes and suggests future research directions.

Chapter 2

A Reasoning Architecture for Exploratory-Corrective Domains

Whether for exploration or repair, exploratory-corrective agents will need to reason "diagnostically" to characterize world states, as well as their objectives. Since action is likely to be necessary for both diagnostic and corrective tasks, such agents will also have to possess some planning capabilities. Here, I will first argue that diagnostic and therapeutic *action* must often be interspersed in diagnosis-and-repair domains. Then, I will present an agent's internal architecture in which the overall *reasoning* task is decomposed into that which is diagnostic in nature and that which is concerned with actions (i.e. planning). Throughout its mission (in our case, throughout a diagnosis-and-repair session), the agent will interleave diagnostic and therapeutic activities by cycling between these two reasoning sub-components and an action/perception module (or, in our case, by interacting with the physician).

First, from JR's case, it is evident that in exploratory-corrective domains such as trauma management it is often hard to separate exploratory activity from activity aimed at repair, whether conceptually or operationally. In particular

- Exploration and repair cannot be temporally separated. For example, faced

with multiple potential injuries, it is often impossible to first explore, and only then act to achieve corrective goals. Urgent repair may have to precede further exploration whereas diagnostic activity may be useful during and after repair for monitoring purposes and to verify actual results;

- The distinction between exploratory and corrective activity may be blurred, and is often based on *intention* rather than actual effect. For example, in JR's case it was demonstrated that a diagnostic test may have a certain therapeutic effect and that, on the other hand, the response to a therapy may have some diagnostic importance;

- Most importantly, exploration and repair are very much co-dependent. On one hand, the decision whether, when, and what to explore, to which extent, and using what means, clearly depends on which repair goals have been identified. On the other hand, those very repair goals can only be determined through exploration.

Thus, since one may have to act to repair before all exploration is concluded, diagnostic reasoning may be put on hold until after this corrective activity is concluded. Another reason to intersperse diagnostic reasoning and planning is that the former may require information that can only be acquired via action. From the other side, corrective action may require reasoning of a diagnostic nature throughout: first to determine the goals that lead the planner to select it, and then to monitor intermediate and final response and verify goal satisfaction from actual evidence. The mutual dependency between exploratory and corrective activities is yet another reason why an exploratory-corrective agent must continuously cycle between diagnostic reasoning and planning.

In the *Exploratory-Corrective Management* (ECM) architecture, *reasoning* is decomposed into two components:

1. A diagnostic reasoner which is iteratively charged with characterizing what is true (the diagnosis); what needs to be determined (the diagnostic goals), and what needs to be achieved (the corrective, or therapeutic, goals);

2. A planner which via the selection and respective ordering of actions for multiple concurrent diagnostic and therapeutic goals serves as a mediator between competing diagnostic and therapeutic needs.

Control in the ECM architecture cycles between these two reasoning components and an action/perception component.

In exploratory-corrective domains, both exploration and repair compete for the same resources. In particular, they both compete for action. Planning is a natural point at which to mediate between a variety of exploratory and corrective needs for action. As the output of a diagnostic reasoner, explicitly specified goals provide a uniform representation for both types of needs, and also serve as a natural interface with the planner.

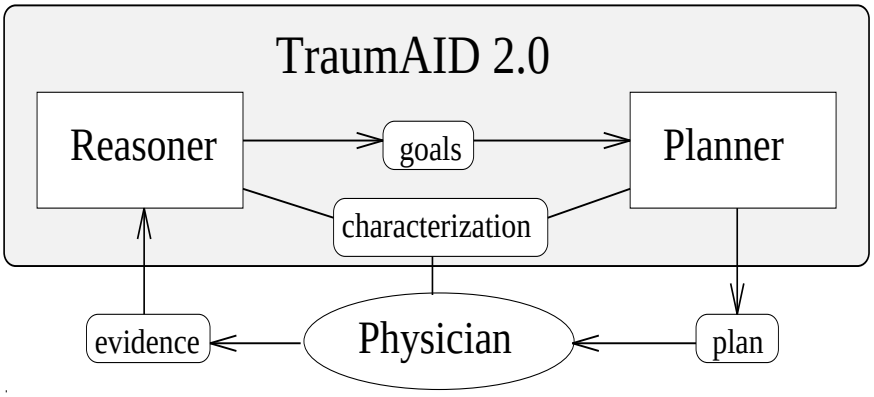


Figure 2.1: ECM architecture: basic cycle of Reasoning, Planning and Action

Figure 2.1 depicts the ECM architecture of TraumAID 2.0 and its interaction with the physician. Reasoning modules are represented in rectangular boxes; rounded rectangles represent various information-types or knowledge used by the system; directed arcs are used to indicate flow of control, whereas undirected ones indicate

simple access. Information types drawn on control-flow arcs are the *architectural duty* of the preceding module, and the triggering input of the subsequent one.

TraumAID 2.0's ECM architecture employs a basic cycle of reasoning and action which is repeated after each action and/or after new information is acquired and reported by the physician. A cycle begins as initial information (evidence) is provided by the physician. From this evidence, the reasoner draws conclusions as to what problems the patient may have, and proposes a set of diagnostic and therapeutic goals. Importantly, goals and not conclusions, recommendations and not mere characterization, are the reasoner's architectural duty. In its turn, the planner decides *how* to address each of the proposed goals and in what *order*. By selecting means to address the current combination of goals, and by scheduling them to reflect urgencies, priorities and other constraints, the planner is effectively mediating between the various needs. The plan it produces is then presented to the physician who is expected to carry out its first action, or first few actions, and then report back results, as well as any relevant evidence that becomes available. A new cycle of diagnostic reasoning, planning and action is then initiated with this new evidence. In this new cycle, the characterization of the current state and the current set of goals may both change to reflect new information and the results of recent actions. A new plan is then constructed to reflect this change.

Given its structure as a closed-loop control system, and given its particular decomposition of the reasoning task, the ECM architecture satisfies the following desiderata deemed necessary by Section 1.2's analysis of JR's case:

1. It allows interleaving diagnosis and repair;

2. In the beginning of each cycle, it allows the diagnostic reasoner to set diagnostic and therapeutic goals; in the end of each loop, it allows it to monitor actions and other events, verify *actual* goal achievement, and adapt goals as necessary;

3. It positions the planner to mediate between concurrent diagnostic and therapeutic needs, and also to dynamically adapt plans according to changes in goals and/or other knowledge.

Next, in Chapters 3-4, I will instantiate the ECM architecture with suitable diagnostic reasoning and planning sub-components.

Chapter 3

A Goal-Directed Formulation of Diagnostic Reasoning

3.1 Overview

Goal-Directed Diagnosis (GDD) is a formalization of the diagnosis task which begins from the principle that *diagnosis is only worthwhile to the extent that it has the potential to affect subsequent repair decisions*, and similarly that *repair is only worthwhile to the extent that it can positively affect the eventual outcome*. This principle is essentially a qualitative analogue of a utility-maximization approach to diagnosis. Per my claims in this dissertation, the key to the GDD logic-based formalization is that explicit representation and reasoning about goals can facilitate *focusing* on worthwhile exploratory-corrective activity. I therefore refer to this principle as the *GDD principle*.

Keeping to the common view of a *diagnosis object* as a characterization of what is true, I augment the *diagnostic task* with a requirement that goals be characterized as well. To address this requirement, throughout a diagnosis-and-repair session, the GDD reasoner aims to establish and maintain both a *belief* (a characterization of

what is true), and an *attitude* (an encoding of relevance) towards the achievement of certain knowledge and/or physical states.

This chapter is organized as follows. Section 3.2 overviews related work. Then, Section 3.3 describes the basic GDD framework; Section 3.4 extends the framework to handle multiple, possibly contradictory, sources of information; Section 3.5 presents a view of the diagnosis process as search in a space of attitude-belief assignment; and Section 3.6 discusses goal inhibition. Section 3.7 exemplifies use of GDD and its various features in a simplified version of TraumAID 2.0's strategy for the diagnosis and treatment of a tension pneumothorax. Section 3.8 outlines a potential extension of GDD with default rules. Finally, Section 3.9 discusses the implementation of GDD in TraumAID 2.0.

3.2 Related Work

> **Diagnosis** (Oxford Dictionary) *Determining the nature of (esp. a disease) from observations of symptoms.*
>
> **Diagnosis** (Webster Dictionary) *1: the art or act of identifying a disease from its signs and symptoms. 2: investigation or analysis of the cause or nature of a condition, situation, or problem (diagnosis of engine trouble).*
>
> **Diagnosis** (Dorland's Illustrated Medical Dictionary) *1: The art of distinguishing one disease from another. 2: The determination of the nature of a case, or a disease.*

In line with the above definitions, it is common to take a diagnosis *object* to be a characterization of a state of affairs, e.g. the determination of the exact problem in a faulty device. The diagnostic *task* is thus commonly taken to be the determination of such a characterization.

Diagnostic reasoning has been an important area of Artificial Intelligence, and has resulted in a number of methodologies and techniques. This large body of research can be categorized along many dimensions; for example

- the *number of faults* subject to diagnosis; especially whether or not a single fault assumption is made;
- the *number of alternative explanations sought*; in particular, whether a single explanation (e.g. most plausible, or most probable) is sought, or is it the case that all possible hypotheses that explain, or even are just consistent with the observed behavior, are of interest;
- *what is modeled*: whether an explicit ("deep") model of the diagnosed system is used, or is it a model, or imitation of the behavior of, a human diagnostician;
- *how it is modeled*: whether formal methods are used (e.g. based on logic and/or probability), or is it a model in which proven domain-specific heuristics are encoded;
- *whether or not observations are actively sought*: some paradigms are concurrent, that is they try to work from a given set of observations, whereas others are sequential and so guide the user in acquiring necessary information;
- *the assumptions made about the state of the system and/or the fault*: some assume a static system, whereas others model change over time, or at least allow for such change; some assume the fault is static whereas others allow for non-monotonic or intermittent faults.

Considering these attributes, the GDD framework is aimed at multiple fault situations. It is a discrete logic-based formalization of an expert's reasoning. While itself a concurrent framework in which one explanation is sought, viewed in conjunction with the ECM architecture it is used for active sequential diagnosis *and* repair.

It aims at a single explanation which is continuously revised. While it allows for non-monotonicity, it lacks explicit reasoning about change.

Next, I overview related formalizations of diagnosis, taking the perspective of an exploratory-corrective domain. I will then discuss AMORD, an early Artificial Intelligence system in which goals are made explicit in order to facilitate better control of the system's forward-chaining inference engine.

3.2.1 Related Formulations of the Diagnostic Task: An Exploratory-Corrective Perspective

Recent years have seen significant advances in approaches for *formalizing* diagnosis, with various formalizations differing along the dimensions just mentioned. In particular, two distinctions which are currently regarded by the community as very important are:

1. *what is formalized*, i.e. is it *human expertise* that is modeled, or is it the function of the diagnosed device (*model-based* [29]); and

2. *the formalization method*, i.e. discrete (e.g. rules, consistency-based, abductive, etc), or probabilistic (e.g. Bayesian classification, influence diagrams, decision- or utility-theoretic, etc.).

What is common to most all of these new formalizations is that they all take a *diagnosis object*, roughly defined as some characterization of the current state of affairs, as a solution and therefore as their objective. In formalizing GDD, I take the view that the *recommendations that result from a diagnosis* are important. Thus, characterization of the appropriate goals, and not merely of what is true, is taken to be the objective of a GDD reasoner.

An important observation made by Poole and Provan [120] is that the optimality of a diagnosis must depend on *post-diagnosis* goals. Provan and Poole [124] advocate

the use of *utilities*. In [121], Poole and Provan further note that there is often no need for a complete explanation and that the *granularity* of a solution depends on its uses (and also on available tests). A similar observation is made by Leake in his work on explanation evaluation [95]: "An explainer's reasons for explaining have profound effects on the information that a good explanation must provide." Leake presents a taxonomy of *explanation purposes* and shows how each one results in a different explanation of the same anomalous event.

In some domains, explicitly specified utilities can be used to structure diagnostic process decisions. However, while utility-theory is increasingly used in Artificial Intelligence in general and medical decision making in particular, it also requires a level of completeness and precision in characterizing a domain that is often impossible or very hard to obtain. The GDD principle can be viewed as a qualitative analogue of a utility-maximization principle. The GDD framework supports the implementation of utility maximizing behavior in ECM agents by explicitly representing and reasoning about goals. In particular, alternative actions can be ruled in or out depending on interactions among competing goals and among alternative actions for addressing these goals, e.g. suppression, subsumption, compatibility, preferences, etc.

I have argued that diagnosis and repair are often inseparable in exploratory-corrective domains. GDD is therefore proposed as part of a *total* approach. Recent work by Friedrich *et al.* [51] and Sun and Weld [149] shares much of this view. Friedrich *et al.*'s theory, for example, has no explicit notion of a diagnosis object. Instead, a sequence of tests and repair actions is sought that if applied to the current state will imply (as in a logical proof) a restoration of the diagnosed system to a proper working condition. Presented not as a theory of diagnosis but as a theory of *repair planning*, their work applies a possible-models planning approach (Winslett, [161]) to a diagnostic domain. Friedrich and Nejdl [52] describe a set of algorithms for diagnosis-and-repair plans. Sun and Weld use UWL, a STRIPS-like language, in an approach which integrates GDE-style diagnosis and STRIPS-style planning. The link

between diagnosis and repair planning in *real* applications is also emphasized by Pepper and Kahn [118]. Also related, although to a lesser extent, is work by Rushby and Crow [132] who formalize reconfiguration, a form of repair, using an extension of Reiter's theory of diagnosis [131]. In GDD, goals are used to focus on repair-worthy issues. In the ECM architecture, *direct* interactions between goals can be resolved using GDD rules (goal-level resolution); *indirect* interactions between goals are resolved by the accompanying planner (see Chapter 5).

An agent may often have to *act* in order to obtain diagnostic information. An important observation from the multiple trauma management domain is that such diagnostic activity may affect the actual state of the patient and not only our knowledge about it. As evident in JR's case, in some instances the very condition a given diagnostic action attempts to diagnose may be affected by it. In most sequential diagnosis frameworks, the decision to act is based on the potential, or expected, discriminatory power of a given piece of information, with little or no consideration to the potential ramifications of the diagnostic activity. Some sequential frameworks use a *cost* measure to model the effect of actions. However, as noted by Genesereth [54], this approach is way too simplistic. In this work, rather than worrying about more accurate models of action, I claim that if properly integrated diagnostic research can simply rely on planning research. Goals, diagnostic and therapeutic, are the architectural duty of the GDD reasoner and serve as a natural interface with an accompanying planner.

3.2.2 AMORD

AMORD [32] is a general purpose problem solver where a reasoning component is accompanied with truth-maintenance and planning facilities. The main thesis behind AMORD's reasoning component is that combinatorial forward-chaining can be avoided via meta-reasoning, i.e. if the problem solver reasons explicitly about its own reasoning strategy. In particular, AMORD's reasoner posts *inference goals* to itself.

These goals, distinct from those posted to its planner, serve to *control* reasoning, i.e. to limit or prioritize the potentially vast space of hypotheses considered. By explicitly encoding the reasoner's control, de Kleer *et al.* hope to focus the reasoning on more promising problem-solving avenues and avoid irrelevant or useless inferences. GDD shares this *intuition*.

Technically, given a set of facts or beliefs, both machineries compute new beliefs and goals. The key to the differences between the two are their distinct objectives: AMORD's objective is to control *reasoning* whereas in the ECM architecture the purpose is to control *actions*. AMORD's reasoner distinguishes two types of goals: *inference goals* which are aimed at controlling its own inference, and *action goals* which are simply posted to the planner. GDD is *not* concerned with inferences at all; whenever it is invoked, it carries out a *complete* inference[1]. It only posts *action goals*, which are then addressed by the system's planner. Interestingly enough, some of these action goals are aimed at *knowledge*, much like AMORD's inference goals. The distinction, however, is that diagnostic (knowledge) goals are only posted when *action* is required.

Key to encoding *strategies* in AMORD is that goals are inferred from facts, and from other goals. In the ECM architecture, while some of the relations between goals (e.g. sub-goal generation, suppression, inhibition) are encoded in GDD rules, much of the interaction between goals (i.e. the overall *strategy*) is resolved by the planner. As further explained in Chapter 5 goals are used to represent choice points (with respect to alternative actions) in local strategies and to facilitate the planner's mediation and coordination between a *combination* of strategies. Furthermore, this is done *implicitly*, i.e. on the fly, and based on general principles.

[1] As a side note, when appropriate, computation can be *represented* as activity and thus be considered explicitly. This is a major area of current research, e.g. [133], but beyond the scope of my current work.

3.3 The Basic Framework

3.3.1 Underlying Framework: Multi-Valued Logics (MVL)

Ginsberg's Multi-Valued Logics (MVL) is a formal framework for inference in which each proposition is assigned not only a truth value, corresponding to the strength of belief in that proposition being *true or false*, but also a knowledge assessment, measuring roughly the amount of knowledge used in deriving this belief [58]. Bilattices in which one partial order corresponds to the truthfulness measure and the other to the knowledge assessment are used in MVL as domains.

My first formalization of GDD [135] used a three-valued logic: *true, false and unknown*. Here, I present an MVL-based reformulation of GDD, the immediate result which is the ability to consistently extend the inference paradigm to more expressive domains. In Section 3.4, I show it to be useful in explicitly representing and reasoning about contradictory information. In Section 3.8, I outline an extension which facilitates reasoning with defaults.

In this new formalization of GDD, each proposition is assigned a value drawn from the cross product of *two* bilattices: one representing belief, the other representing attitude. The notion of belief is interpreted regularly, representing one's belief in whether a certain propositional statement is true or not. The attitude component is used to represent problem-solving control information and measures the relevance of acquiring information about, or achieving the condition described by, the particular proposition. The belief bilattice still has the truthfulness and knowledge partial orders whereas the attitude bilattice has relevance and knowledge dimensions respectively.

Within Ginsberg's paradigm, domain knowledge is expressed by general first-order formulae, and thus requires an underlying theorem prover. In contrast, TraumAID 1.0 used a rule-based representation. In formalizing TraumAID 2.0, rather

than adopting Ginsberg's formulation in full, and then having to completely reconstruct the domain knowledge, I have specialized MVL to the rule-based case. Besides fitting my needs, I believe that the specialized theory may be useful for others wishing to convert other rule-based systems to MVL. While the material presented next is self-contained, the reader is referred to [58] for a more complete coverage of MVL.

3.3.2 Attitude and Belief

During the diagnostic process, the GDD reasoner will maintain and update an *attitude* and a *belief* for a set of propositional statements. To remain general, propositions may be any fact about the patient or the world that the reasoner may know to hold, may know not to hold, may assume, may want to know whether it holds, may want to achieve, may be confused about, etc. As just mentioned, the problem solver's belief in whether a proposition holds is separated from its attitude towards acquiring this knowledge. For example, the perceived relevance of determining whether or not a patient suffers from a *pericardial tamponade*[2] is separated from the current belief about the patient's condition. As expected, the problem solver's attitude towards and/or belief in a given proposition will change over time as a result of new information becoming available, new inferences being drawn, activity being carried out, etc.

Definition 3.3.1 *A Lattice*

A *lattice* is a triple (L,$\wedge$,$\vee$), where L is its domain, $\wedge$ ("meet") and $\vee$ ("join") are binary operations from L onto itself that are

1. idempotent, i.e. $a \wedge a = a$, $a \vee a = a$;

2. commutative, i.e. $a \wedge b = b \wedge a$, $a \vee b = b \vee a$;

[2] A pericardial tamponade is a condition in which blood fills the pericardial sac, interfering with the heart's operation.

3. associative, i.e. $(a \wedge b) \wedge c = a \wedge (b \wedge c)$, $(a \vee b) \vee c = a \vee (b \vee c)$; and

4. obey the absorption laws, i.e. $a \wedge (a \vee b) = a$, $a \vee (a \wedge b) = a$.

Alternatively, a lattice can be defined as a partially ordered set (poset), any two elements of which have a greatest lower bound (*glb*) and a least upper bound (*lub*). The two definitions coincide by taking $lub(a, b) = a \vee b$, and $glb(a, b) = a \wedge b$. A lattice is said to be *complete* if *lub* and *glb* can be defined for any subset of the lattice's elements. (In particular, a complete lattice has minimal and maximal elements.) Thus, any lattice with a finite domain is complete.

Definition 3.3.2 *A Bilattice*

A *bilattice* is a sextuple $(B, \wedge, \vee, \cdot, +, \neg)$ such that:

1. $L_1 \stackrel{\text{def}}{=} (B, \wedge, \vee)$ and $L_2 \stackrel{\text{def}}{=} (B, \cdot, +)$ are both complete lattices; and

2. $\neg : B \rightarrow B$ is a mapping such that:

   (a) $\neg^2 = 1$; and

   (b) $\neg$ is a lattice homomorphism from L_1 to its dual $\overline{L_1} \stackrel{\text{def}}{=} (B, \vee, \wedge)$ and from L_2 to itself.

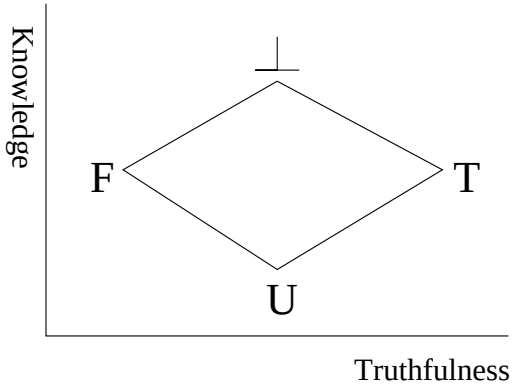


Figure 3.1: Basic Truth-Knowledge Bilattice

Ginsberg discusses bilattices that are based on two partial orders: truth-wise ($\leq_t$), and knowledge-wise ($\leq_k$). That is, each proposition is described by how strongly we believe it is true, and by how much knowledge was involved in inferring this belief. The smallest nontrivial bilattice (Figure 3.1) has four points: T (absolute truth), F (false), U (unknown), and $\perp$ (contradictory). In that bilattice, the $\leq_t$ partial order defines one lattice (B,$\wedge$,$\vee$), whereas the $\leq_k$ defines another lattice (B,$\cdot$,+). In any truth-knowledge bilattice, negation reverses the truth capacity of a proposition, leaving its knowledge capacity unchanged. In particular, within the basic bilattice, $\neg$ maps T to F and vice versa, leaving U and $\perp$ untouched.

Two, possibly distinct[3], bilattices are used in GDD as domains for attitude and belief respectively:

Definition 3.3.3 *Attitude and Belief*

Given a set of primitive propositions $\mathrm{H} \stackrel{\mathrm{def}}{=} \{h_i\}_{i=1}^{n}$,

- an *attitude* maps H to an attitude bilattice B_A
- a *belief* maps H to a belief bilattice B_B
- an *attitude-belief* combines the two and maps H to the cross product $B_A \times B_B$. Alternatively, it can also be viewed as a pair $\langle \phi_A, \phi_B \rangle$ of attitude and belief.

For the most of this paper, we will presume both B_A and B_B to be 4-point bilattices such as those in Figure 3.2. The belief bilattice, following Ginsberg's suggestion, is defined by the truth-knowledge partial orders. In the attitude bilattice, a proposition is described by its relevance ($\leq_r$) and the knowledge used to derive this relevance ($\leq_k$). Note that one's knowledge with respect to the truthfulness of a proposition need not equal one's knowledge with respect to the relevance of that same proposition. The extreme points in the belief bilattice are labeled T and F, whereas same

[3]Given the features of a certain domain, it may conceivably be useful to choose an attitude bilattice that is more or less expressive than the belief bilattice.

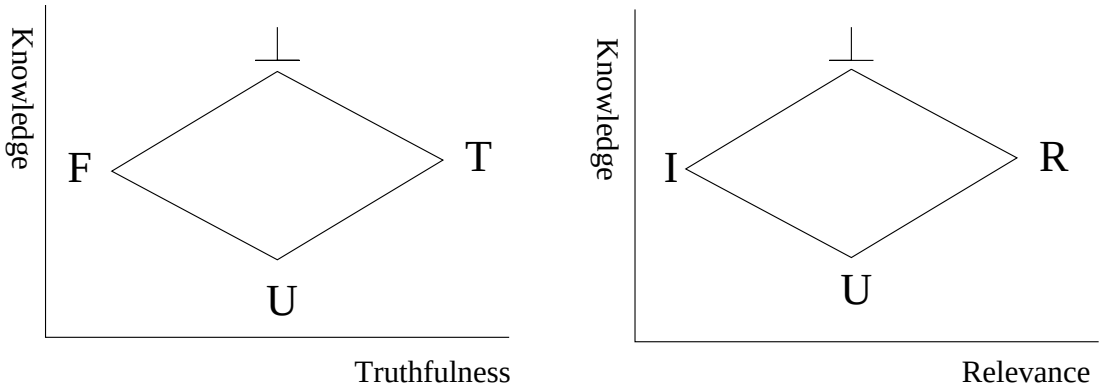


Figure 3.2: Four-point Belief and Attitude Bilattices

points are labeled R (for relevant) and I (for irrelevant) in the attitude bilattice. Extensions to more complex bilattices, in which defaults can be represented, will be discussed in Section 3.8.

3.3.3 The Interpretation of an Attitude-Belief Assignment (and What Are Goals)

Throughout all stages of a given diagnosis-and-repair session, an attitude-belief value will be assigned to all propositions known to the system. This attitude-belief assignment will then be updated whenever, and in accordance with, any new information which becomes available.

Semantically, the belief assignment to a given proposition is interpreted as an assessment of whether or not we believe the statement denoted by this proposition holds or not. For example, if ϕ_B(`Hemothorax`)=T, then we believe that the patient suffers from a hemothorax. Similarly, if ϕ_B(`Hemothorax_is_Relieved`=T, then we believe that the hemothorax has in fact been relieved.

The notion of "goal" is derived from the attitude assignment to a given proposition. Generally speaking, a proposition p is a goal if its *attitude* assignment is high on the relevance partial order (in our 4-point bilattice if $\phi_A(p)$=R). Note that there is no *syntactic* distinction between a diagnostic (knowledge seeking)

and a therapeutic (state achieving) goal. For example, the relevance of *knowing* whether or not a patient suffers from a hemothorax is indicated via the attitude assignment ϕ_A(`Hemothorax`)=R and similarly the relevance of *achieving a state* in which the hemothorax is achieved is indicated via the attitude assignment ϕ_A(`Hemothorax_is_Relieved`)=R.

Of course, not every relevant proposition becomes automatically an *operational* goal. For example, if ϕ_B(`Hemothorax`)=T then it is already believed that the patient does suffer from a hemothorax and thus, as a knowledge goal, it is already satisfied. An important distinction between a diagnostic and a therapeutic goal is that while the latter is often regarded satisfied when *any* concrete belief can be assigned to the given proposition, the latter is only achieved when a positive determination is made. For example, the `Hemothorax` goal is satisfied when it is assigned a belief value of either T or F, whereas the `Hemothorax_is_Relieved` goal is only satisfied when it is assigned a belief value T.

In general, in more complex bilattices, one has to define which combinations of relevance and achievement levels need be addressed, in what order of preference, etc. However, since its inference is purely syntactic, the GDD reasoner is indifferent to these semantic subtleties. Thus, in the ECM architecture, it is the planner's role to decide which goals to pursue.

3.3.4 Knowledge Representation

GDD uses rules to represent knowledge. Two types of rules are used: one for inferring belief, the other for inferring attitude. Antecedents in both types of rules are stated in belief *terms*, the interpretation of which is soon to be specified. A rule's consequent (header) must be either a proposition p or its negation $\neg p$. An inference procedure, soon to be presented, will specify the way in which belief and attitude assignments are inferred for a rule's consequent given its antecedents.

Definition 3.3.4 *An Antecedent*

An *antecedent* is recursively defined as either:

1. a proposition h;
2. *relevant*(h), where h is a proposition;
3. $\neg a$, where a is an antecedent;
4. *true*(a), *false*(a), *unknown*(a), or *contradictory*(a), where a is an antecedent; or
5. *known*(a), *unless*(a), or *compatible-with*(a), where a is an antecedent.

Definition 3.3.5 *Rules*

A *rule* R has two parts: a *body*, or a premise, which is a set of antecedents, and a *header* (or consequent) which is a single proposition. Let Ant_i denote an antecedent, and let d denote a consequent. GDD has two types of rules:

1. *Evidential rules* are used to define and compute belief. They map evidence and lower-level conclusions to conclusions and take the form:

$$Ant_1 \wedge Ant_2 \wedge \ldots \wedge Ant_r \Rightarrow d \quad .$$

   For example, the following evidential rule serves to conclude a possible ischemic spinal cord injury:

$$\begin{array}{l} \textit{Loss\_of\_motor\_in\_legs} \wedge \\ \textit{Thoracic\_aortic\_injury} \wedge \\ \neg \textit{Compound\_fracture\_vertebra} \\ \qquad\qquad \Rightarrow \textit{Ischemic\_spinal\_cord\_injury} \end{array}$$

2. *Goal Setting rules* are used to define and compute attitude. They map evidence and conclusions to attitude and take the form:

$$Ant_1 \wedge Ant_2 \wedge \ldots \wedge Ant_r \ \triangleright \ d.$$

For example, the following goal-setting rule concludes whether it is relevant to know whether a patient has hematuria[4].

$$\begin{array}{l} \mathit{Gunshot\_wound\_to\_abdomen} \wedge \\ \mathit{Bullet\_in\_abdomen} \\ \qquad\qquad \triangleright\ \mathit{Hematuria} \end{array}$$

A fact may often be provable in several *alternative* ways. Similarly, a goal may need to be set in a variety of contexts. The same conclusion can therefore be drawn through different evidential rules, and the goal can be set through different goal-setting rules.

Conversely, a proposition p can head different goal-setting *and* evidential rules. In particular, a goal-setting rule headed by p is used to convey that it is worth acquiring knowledge about p or that it is worth achieving a state in which p holds (depending on p's semantic interpretation as a diagnostic or repair goal). An evidential rule headed by p is used to conclude whether or not it holds, or put differently whether or not it has been satisfied.

Example 3.3.6 Consider the diagnosis and repair of a pericardial tamponade. Throughout this process, the following diagnostic and therapeutic goals are instantiated, addressed, and satisfied:

1. **Rules for setting a diagnostic (knowledge) goal:**

   The need to investigate a pericardial tamponade will first be triggered by such observations as muffled heart sounds or observable injury to the sternum, and rules of the form:

   $(\ldots) \triangleright \mathit{Pericardial\_Tamponade}$

   "It is relevant to *know* if the condition holds".

[4]Hematuria is a condition in which blood appears in the urine, probably indicating some internal bleeding.

2. **Rules for satisfying a diagnostic (knowledge) goal:**

   This diagnostic goal will then be addressed by the planner, recommending an ultrasound effusion or a needle aspiration of the pericardial sac depending on the patient state. Such test will then be incorporated into the overall plan, considering other concurrent needs as well. Results from such test will then serve to actually determine whether or not the patient suffers a pericardial tamponade, using rules of the form:

   $(\ldots) \Rightarrow \mathit{Pericardial\_Tamponade}$

   "Conclude whether the condition holds".

3. **Rules for setting a therapeutic goal:**

   If the patient is determined to suffer a pericardial tamponade then, possibly depending on other conditions s/he may suffer, a therapeutic goal of treating the condition may be set, using rules of the form:

   $(\ldots) \blacktriangleright \mathit{Relieve\_pressure\_pericardial\_sac}$

   "It is relevant to *address* the condition".

4. **Rules for satisfying a therapeutic goal:**

   The *actual* results of the treatment can then be used to determine whether or not it was successful, using rules of the form:

   $(\ldots) \Rightarrow \mathit{Relieve\_pressure\_pericardial\_sac}$

   "Conclude whether the condition has been successfully addressed".

Note again that there is *no syntactic* distinction between the encoding of rules used to set diagnostic or knowledge goals and conclude that they are satisfied and rules used to encode the same for therapeutic, or state achieving goals. The sole distinction between diagnostic and therapeutic goals is a semantic one. A goal's nature affects what is *done* when it is set, and the conditions under which it can be regarded satisfied. In the ECM architecture, both are to be determined by the planner.

While rules used by GDD to express knowledge have their antecedents expressed solely in belief terms, it may often be useful to predicate whether a proposition is a goal based on the relevance (or irrelevance) of another goal. To facilitate this within the belief-based antecedent calculus which will soon be introduced, a mapping can be added from the attitude bilattice to the belief bilattice (denoted *attitude-to-belief*), roughly modeling the belief in the relevance of a given proposition.

$$\textit{attitude-to-belief}(a) \stackrel{\text{def}}{=} \begin{cases} T & a = R \\ F & a = I \\ a & \textit{otherwise} \end{cases}$$

This mapping is clear when B_A and B_B are isomorphic. However, in general, one has to make sure that relevance information is not lost in the transformation (e.g. if the relevance bilattice was chosen to have a finer granularity than the belief bilattice).

Definition 3.3.7 *A Diagnostic Problem*

A *diagnostic problem* is a quadruple $P \stackrel{\text{def}}{=} \langle \text{H,RB,M}_0\text{,OBS} \rangle$ such that:

- $\text{H} = \{h_1, h_2, \ldots, h_n\}$ is a set of propositions;
- RB is a set of evidential and goal-setting rules;
- $\text{M}_0 \subseteq \text{H}$ is a set of observations;
- $\text{OBS} : \text{M}_0 \rightarrow \text{B}_B$, is a partial belief function.

3.3.5 Solving a Diagnostic Problem

Solving a diagnostic problem will require computing an *inferential closure* of the observations given the evidential and goal-setting rules. The basic form of inference, presented next, is simpler and more intuitive than that in [58]. In Section 3.4, I extend it to a form that is closer to Ginsberg's in order to support reasoning about

contradictory information. Before defining the basic inferential closure, I first define the semantic interpretation of rules:

Definition 3.3.8 *Belief Assignment for an Antecedent*

Let $\langle \phi_A, \phi_B \rangle$ be the current attitude-belief (defined on propositions). For any antecedent Ant, the inferred belief in that antecedent, $\phi_B^*(Ant)$, is defined as follows:

1. If Ant is a proposition h, $\phi_B^*(Ant) \stackrel{\text{def}}{=} \phi_B(h)$;
2. If Ant is of the form *relevant*(h), $\phi_B^*(Ant) \stackrel{\text{def}}{=}$ *attitude-to-belief*$(\phi_A(h))$;
3. If Ant is of the form $\neg Ant'$, where Ant' is an antecedent, $\phi_B^*(Ant) \stackrel{\text{def}}{=} \neg\phi_B^*(Ant')$;
4. If Ant is of the form $f(Ant')$, where Ant' is an antecedent, then
   - If f is a *belief predicate* (*true, false, unknown, contradictory*), then

$$\phi_B^*(true(Ant')) \stackrel{\text{def}}{=} \begin{cases} T & \phi_B^*(Ant') = T \\ F & otherwise \end{cases}$$

$$\phi_B^*(false(Ant')) \stackrel{\text{def}}{=} \begin{cases} T & \phi_B^*(Ant') = F \\ F & otherwise \end{cases}$$

$$\phi_B^*(unknown(Ant')) \stackrel{\text{def}}{=} \begin{cases} T & \phi_B^*(Ant') = U \\ F & otherwise \end{cases}$$

$$\phi_B^*(contradictory(Ant')) \stackrel{\text{def}}{=} \begin{cases} T & \phi_B^*(Ant') = \bot \\ F & otherwise \end{cases}$$

   - Otherwise
     - $\phi_B^*(known(Ant')) \stackrel{\text{def}}{=} \begin{cases} F & \phi_B^*(Ant') = U \\ \bot & \phi_B^*(Ant') = \bot \\ T & otherwise \end{cases}$
     - $\phi_B^*(unless(Ant')) \stackrel{\text{def}}{=} \neg\phi_B^*(Ant') \vee \neg\phi_B^*(known(Ant'))$
     - $\phi_B^*(compatible\text{-}with(Ant')) \stackrel{\text{def}}{=} \phi_B^*(Ant') \vee \neg\phi_B^*(known(Ant'))$

Belief predicates map belief-terms to {T,F} and are useful where reasoning in absolute terms is necessary. Examples of this are presented in Sections 3.4.4, 3.6, and 3.8. Thus, note that h and *true(h)* are interpreted differently and the same is true of $\neg h$ and *false(h)*. Similarly, $\neg$*known(h)* refers to any situation in which a concrete belief cannot be reached whereas *unknown(h)* describes the particular situation in which h is assigned a belief value U, denoting *lack of information*. In the basic bilattice, $\perp$ is also categorized as $\neg$*known(h)*; in Section 3.4, I will use $\perp$ to denote unknown due to co-presence of contradictory information.

Definition 3.3.9 *Belief Assignment for a Rule's Body*

Let $\langle \phi_A, \phi_B \rangle$ be our current attitude-belief, and let R be a rule with a collection $\{Ant_i\}_{i=1}^{k}$ of antecedents in its body. We will interpret a rule's body as a simple conjunction by defining:

$$\phi_B^*(\text{body(R)}) \stackrel{\text{def}}{=} \wedge_{i=1}^{k} \phi_B^*(Ant_i)$$

Definition 3.3.10 *Consistent Inference*

An attitude-belief $\langle \phi_A, \phi_B \rangle$ is a *consistent inference* for a diagnostic problem P iff

1. It coincides with OBS, i.e for all h$\in M_0$ ϕ_B(h)=OBS(h);

2. For any proposition d$\in$H–M$_0$, let $\{R_i\}_{i=1}^{l}$ be the set of evidential rules with d as their header and let $\phi_B^*(R_i)$ denote the evaluation of R_i's body as in Definition 3.3.9, then these rules are interpreted disjunctively:

$$\phi_B(d) = \vee_{i=1}^{l} \phi_B^*(R_i);$$

3. Similarly, for any proposition d$\in$H, let $\{R_i\}_{i=1}^{l}$ be the set of goal-setting rules with d as their header and let $\phi_B^*(R_i)$ denote their evaluation, then

$$\phi_A(d) = \textit{attitude-to-belief}^{-1}(\vee_{i=1}^{l} \phi_B^*(R_i));$$

Example 3.3.11

1. Consider the evidential rule in Definition 3.3.5. If
   $\phi_B(\texttt{Loss\_of\_motor\_in\_legs})$=T
   $\phi_B(\texttt{Thoracic\_aortic\_injury})$=T, and
   $\phi_B(\texttt{Compound\_fracture\_vertebra})$=U
   then
   $\phi_B^*(\neg\texttt{Compound\_fracture\_vertebra})= \neg\phi_B(\texttt{Compound\_fracture\_vertebra})$=U
   and therefore
   $\phi_B^*(\texttt{Ischemic\_spinal\_cord\_injury})$ = T$\wedge$T$\wedge$U=U

2. Consider the goal-setting rule in Definition 3.3.5. If the rule's antecedents evaluate to T, then
   $\phi_B^*(\texttt{Hematuria}) =$ *attitude-to-belief*(T)=R

Solving a diagnostic problem requires computing a consistent inference. While in general, there is no guarantee that such an inference is unique or computable, or that it even exists, the following straightforward (and thus inefficient) procedure has worked so far.

Algorithm 3.3.12 *Computing an Inferential Closure*

1. Start off with the observations, by setting
   $\phi_A(\text{h}) \stackrel{\text{def}}{=}$ U, for all h$\in$H $\phi_B(\text{h}) \stackrel{\text{def}}{=} \begin{cases} OBS(h) & h \in M_0 \\ U & otherwise \end{cases}$
2. Forward chain on the rules, enforcing conditions 2 and 3 of the Consistent Inference definition (Definition 3.3.10), until reaching a fixed point.

It is the formal definition of a *solution* in the GDD framework which differentiates it from other formulations of diagnosis. Let us first define a *diagnosis*:

Definition 3.3.13 *A Diagnosis*

Let $\langle \phi_A, \phi_B \rangle$ be an inferential closure for a problem P, then ϕ_B, its belief part, is a *diagnosis* for P.

Indeed, most formalizations of diagnosis take a *diagnosis* as their solution. In GDD, however, we associate more importance with the goals (and consequently the actions) adopted during the diagnosis process. We shall therefore define a solution as follows:

Definition 3.3.14 *A Solution to a Diagnostic Problem*

A *solution* to a diagnostic problem P is the *complete* inferential closure $\langle \phi_A, \phi_B \rangle$.

Consider, for example, the situation in Example 3.3.11 where it is unknown whether the patient suffers an ischemic spinal cord injury. The *diagnosis* is thus deficient in that regard. However, if according to the *goal-setting* rules for this particular situation, ϕ_A(`Ischemic_spinal_cord_injury`)=I (irrelevant), or even U (unknown), then it may not be relevant and the *solution* is thus satisfactory.

In the ECM architecture, solving the current diagnostic problem has an operational purpose too: it defines the goals to be pursued next by the planner and hence the relative importance of goals. In the ECM algorithm (Algorithm 3.3.15), the GDD reasoner is invoked whenever new information is acquired. The solution it provides is subsequently used to guide the planner in the choice of activity which, in turn, may provide new information to start a new cycle.

Algorithm 3.3.15 *Diagnosis-and-Repair in the ECM Architecture*

1. Initialize $\langle \phi_A, \phi_B \rangle$ to coincide with OBS;
2. Compute an inferential closure for $\langle \phi_A, \phi_B \rangle$;
3. Construct a plan P for the combination of goals indicated by that closure;
4. Until P is empty do
   - Execute P until new evidence arrives;
   - Update $\langle \phi_A, \phi_B \rangle$ to reflect this evidence;
   - Go to step 2;

Note that the termination criterion is not necessarily related to the completeness of of the working diagnosis. The process terminates when the plan is empty, i.e. when all goals have been addressed, or no means are available for addressing remaining goals, etc. On the other hand, while goals are the "architectural duty" of the GDD reasoner, the belief part of the solution (characterization) is still important serving two purposes: first, to human users, it is useful for the system to be able to *explain* its decisions. Second, beliefs can be useful for *planning* separate from the goals they give rise to, as they might affect choice of means and/or ordering decisions.

3.4 Reasoning about Contradictory Information

In this section I first motivate explicit representation and reasoning about contradictory information. Then, I extend the representational language and inference paradigm to support such reasoning.

3.4.1 Motivation

Even though $\perp$ is part of both the belief and the attitude bilattices, it cannot be assigned to a proposition in the framework developed thus far; this follows directly from the fact that propositions are never initialized with $\perp$, the fact that truth assignments are computed using $\vee$ and $\wedge$ only, and the fact that $\perp$ is not in the closure of $\{T, F, U\}$ under $\wedge$ and $\vee$. On one hand, this is a good feature of the representation and inference scheme. Ginsberg, for example, notes that $\perp$ tends to "pollute" an interpretation. That is, when $\perp$ is assigned to one proposition, it will often propagate to many other dependent propositions. On the other hand, I will argue that $\perp$ is important as an explicit record of contradictory evidence and will suggest a few technical ways to overcome the pollution problem.

Consider two alternative methods used by TraumAID 2.0 to diagnose a *tension pneumothorax* (TP):

1. *X-ray_shows_Tension_Pneumothorax(Side=S)*
   $\Rightarrow$ *Tension_Pneumothorax(Side=S)*

2. *Needle_aspiration_shows_pressure_in_chest(Side=S)*
   $\Rightarrow$ *Tension_Pneumothorax(Side=S)*

If one follows a *confirmation strategy*[5] then positive results in *one* test justify a diagnosis. However, what if one test is negative, and the other has not been taken? By the semantic interpretation of the above rules, TP is unknown (computed as F$\vee$U). It is not proven, but cannot be dismissed since not all the rules have failed. On the other hand, in standard medical practice, a negative test suffices to rule out TP so long as no other indication contradicts these test results. Furthermore, it is medically incorrect to perform the other test just for complete vindication. Insofar as this other test is not performed and provides contradictory evidence, TP should be *assumed* absent (note the non-monotonicity). This reasoning may be encoded in

[5] See Section 7.2.

the following augmented GDD rules:

1'. *X-ray_shows_Tension_Pneumothorax(Side=S)* $\wedge$
compatible-with(Needle_aspiration_shows_pressure_in_chest(Side=S))
$\Rightarrow$ *Tension_Pneumothorax(Side=S)*

2'. *Needle_aspiration_shows_pressure_in_chest(Side=S)* $\wedge$
compatible-with(X-ray_shows_Tension_Pneumothorax(Side=S))
$\Rightarrow$ *Tension_Pneumothorax(Side=S)*

With the addition of a new antecedent, each of these rules will now fail if either test is negative. Otherwise, if either or both tests come up positive, and there is no contradictory evidence, TP is positively concluded.

Now consider the case in which the two tests are performed and provide *contradictory* evidence. Given the first set of rules, TP will be concluded. Given the modified set, both rules will fail: one due to the explicit requirement for a positive result, the other due to the compatibility requirement. TP will thus be concluded absent. However, although it may well be the case that TP is absent, in such instances medical practice defaults to treatment *as if* TP was present.

So far, a proposition is true or false if it is so asserted or concluded. It is labeled unknown if it is not reported or cannot be concluded. In other words, unknown arises from *lack* of knowledge. Contradictions belong to a *different class* of unknowns. Commonly arising in practice, while themselves representing an inconsistency (either in what one observes or in one's interpretation of what one observes), it is important to represent and cope with them in a consistent manner. In order to do so, I first extend the rule representation to allow explicit negative conclusions and then generalize the inference accordingly.

3.4.2 Explicit Negative Conclusions

Reconsidering the *original* set of rules for TP, two rules can be added in which TP is *ruled out* whenever either of the two tests indicates its absence:

3. $\neg$*X-ray_shows_Tension_Pneumothorax(Side=S)*
$\Rightarrow \neg$ *Tension_Pneumothorax(Side=S)*

4. $\neg$*Needle_aspiration_shows_pressure_in_chest(Side=S)*
$\Rightarrow \neg$ *Tension_Pneumothorax(Side=S)*

TP is now concluded if either test is positive, while $\neg$TP is concluded if either test is negative. A conflict arises if we want to interpret $\neg$TP as the negation of TP, namely when both tests are taken and one succeeds whereas the other fails. In such instance, both TP and $\neg$TP will be assigned a truth value T. Ginsberg suggests a more general first-order framework under which conflicts can be represented. The extended form of inference presented next is a specialization of his approach to the propositional rule-based case.

3.4.3 Inference

Inference must now be redefined to allow for rules that explicitly falsify a proposition and for the need to *combine* positive and negative contributions. In [58], the full first-order formulation of MVL assigns to a statement a truth-value that combines the results of *all possible proofs* for that statement and its negation. Theoretically, such determination may be undecidable; in practice, it may often be cumbersome. In contrast, in the propositional rule-based case, proofs need not be searched for since they are given explicitly by the corresponding rules.

In the new definition of consistent inference (Definition 3.4.1), the body of each rule headed by a given proposition p or its negation $\neg p$ is first evaluated. The resulting values for p and $\neg p$ are then combined using the + operator. This interpretation

is natural because one can view each rule as a separate argument for or against p. Combining them is thus a synthesis of knowledge sources, and + is the operator that "adds" along the bilattice's knowledge dimension. Since conditions for both p and $\neg p$ are specified explicitly, "negation as failure" is unnecessary. To avoid negation by failure, each rule's body is padded with U. Thus, a proposition p evaluates to F only if at least one of the rules for $\neg p$ succeeds, but not necessarily when all rules for p fail.

Definition 3.4.1 *Consistent Inference (Redefined)*

An attitude-belief $\langle \phi_A, \phi_B \rangle$ is a consistent inference for a diagnostic problem instance P iff

1. It coincides with OBS, i.e for all h$\in M_0$ ϕ_B(h) = OBS(h);

2. For any proposition d$\in$H–M$_0$, let $\{R_i\}_{i=1}^k$ be the set of evidential rules with d as their header and $\{\overline{R}_i\}_{i=1}^l$ the set of evidential rules with $\neg d$ as their header. Let $\phi_B^*(R)$ denote the evaluation of a rule R body as in Definition 3.3.9, then

$$\phi_B(d) = \sum_{i=1}^k (\phi_B^*(body(R_i)) \vee U) + \sum_{i=1}^l (\neg \phi_B^*(body(\overline{R}_i)) \wedge U);$$

3. Similarly, for any proposition d$\in$H, let $\{R_i\}_{i=1}^k$ be the set of goal-setting rules with d as their header and $\{\overline{R}_i\}_{i=1}^l$ the set of goal-setting rules with $\neg d$ as their header. Let $\phi_B^*(R)$ denote a rule's evaluation, then

$$\phi_A(d) = \textit{attitude-to-belief}^{-1}(\sum_{i=1}^k (\phi_B^*(R_i) \vee U) + \sum_{i=1}^l (\neg \phi_B^*(\overline{R}_i) \wedge U));$$

Using the above inference scheme and the four rules for TP, a positive determination (T) will be made if either or both tests come up positive. A negative determination (F) will be made if one test is negative and the other is either negative or has not been taken. Finally, if the two tests contradict each other, then TP will be assigned a $\perp$ value.

3.4.4 Dealing With Contradictions

Two major concerns need to be addressed in representing and reasoning about contradictory information. The first problem concerns the semantic interpretation of a contradiction. Let p be a proposition for which a contradiction ($\perp$) has been inferred. (I describe this in terms of belief, but unless said otherwise, the same applies to attitude.) Consider rules whose antecedents refer to p. The following two cases should be distinguished:

1. Antecedents of the form $f(p)$, where f is a predicate, will either contribute T, if f is the *contradictory* predicate (cf. Definitions 3.3.4, 3.3.8) or F otherwise. Such antecedents do not present a problem since they map $\perp$ back to the more conservative set of values.

2. Antecedents of the form p, $\neg p$, $\neg$*known*(p), *compatible-with*(p), *unless*(p) or *unknown*(p) yield $\perp$ when evaluated on $\perp$, and thus propagate it. This is the "pollution" phenomenon.

Prior to assessing the ramifications of pollution, consider the problem of what to *do* when a proposition is assigned a $\perp$ belief. Unfortunately, the answer varies. In some cases, a positive (or negative) *default* is appropriate. For example, when asked if Tweety can fly, given the two contradictory rules for birds and penguins, a negative default should be preferred. In other circumstances, none of the conclusions can be safely made and further investigation is warranted. When tests have low reliability, for example, a third test may be taken or one of the tests may be repeated. In yet other cases, a conclusive decision may be irrelevant (e.g. vis-a-vis the GDD principle), or a default to a particular treatment may be preferred. For example, when diagnosing TP, one will default to treating *as if* TP was positively diagnosed. Note however the distinction between defaulting to positive conclusion versus defaulting to positive treatment.

The GDD principle frees us from resolving irrelevant contradictions. Otherwise, the *contradictory* predicate can be used to *explicitly* specify the desired behavior. For example, the following rule concludes the need to treat *as if* TP is positive:

$$contradictory(Tension\_Pneumothorax(Side{=}S)) \\ \triangleright\ Rx\_Tension\_Pneumothorax(Side{=}S)$$

As for the pollution problem, it is *insignificant* when the proposition to which a $\perp$ belief was assigned has no other propositions dependent upon it, or when this proposition and its dependents are all irrelevant. Otherwise, a $\perp$ belief must simply be tolerated as a *type of unknown*. If well programmed, pollution of relevant propositions in the ECM architecture is by definition a temporary phenomenon whose resolution is attempted in subsequent cycles. Finally, the spread of pollution can be blocked by padding rules with antecedents of the form $\neg contradictory(p)$ where p is a proposition which may be polluted. This scheme can either be applied across the board (as a macro expansion) or tailored to particular situations.

3.5 Diagnosis-and-Repair as Search

So far, the discussion has been limited to individual diagnostic problems. However, embedded in the ECM architecture, the GDD reasoner will have to solve a number of such problems in each diagnosis-and-repair session. In this section, I discuss possible changes in an agent's attitude-belief through a complete management session. I first portray the diagnosis-and-repair process as a search through a space of mental states, namely attitude-belief assignments. Then, given the GDD principle, I will discuss restrictions on the directions this search can take.

One way to view the overall diagnostic process is as a path-traversal in an Attitude-Belief (AB) space. Each transition on that path corresponds to a change in the reasoner's previous attitude-belief assignment. Such change can either be the result

of new observations, volunteered or acquired, or they can be the result of recent inferences.

Definition 3.5.1 *Updated Attitude-Belief*

Let $\Gamma \stackrel{\text{def}}{=} \langle \phi_A, \phi_B \rangle$ be an attitude-belief. We define $\Gamma\,|_{\phi_B(h)=v}$ to be an attitude-belief that is the same as Γ except that the belief in h is updated to v (where $v \in \mathrm{B}_B$). Similarly, we define $\Gamma\,|_{\phi_A(h)=v}$ to be the same as Γ except that the attitude toward h is updated to v (where $v \in \mathrm{B}_A$).

Definition 3.5.2 *Accessibility Relation in the AB-space*

The *immediate accessibility* relation in the AB-space corresponds to a single change in attitude or in belief with regard to a *single* proposition h. Let $\Gamma \stackrel{\text{def}}{=} \langle \phi_A, \phi_B \rangle$ be a state, h a proposition and v the new attitude or belief for h, then

$$\text{Result}(\Gamma,\ \text{h},\ \text{v}) \stackrel{\text{def}}{=} \Gamma\,|_{\phi_A(h)=v}, \text{ when the change is in attitude; or}$$
$$\text{Result}(\Gamma,\ \text{h},\ \text{v}) \stackrel{\text{def}}{=} \Gamma\,|_{\phi_B(h)=v}, \text{ when the change is in belief.}$$

The *accessibility* relation Result* generalizes Result to a sequence of updates. Given an initial state $\Gamma \stackrel{\text{def}}{=} \langle A, B \rangle$ and a sequence of updates $\{h_i, v_i\}_{i=1}^{n}$, we recursively define

$$\text{Result}^*(\Gamma, \{h_i, v_i\}_{i=1}^{n}) \stackrel{\text{def}}{=} \begin{cases} \mathit{Result}(\Gamma, h_1, v_1) & \mathit{if}\ n = 1 \\ \mathit{Result}(\mathit{Result}^*(\Gamma, \{h_i, v_i\}_{i=1}^{n-1}), h_n, v_n) & \mathit{otherwise} \end{cases}$$

Adopting this convention, a diagnostic session can be mapped to a traversal in the AB-space which starts from an *initial state*, one that describes the initial observations when diagnosis commences, to a *goal state*, one which corresponds to some desired or acceptable state of affairs. Some transitions correspond to inferences made by the GDD inference procedure, whereas others correspond to information that is either volunteered or obtained via action. Importantly, while at a given state, a number of transitions may be *valid*; however, many fewer will also be *worth taking*, i.e. when combined, will reliably and effectively bring the agent to a goal state.

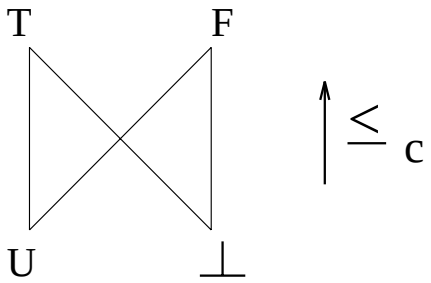


Figure 3.3: Concreteness Partial Order ($\leq_c$)

Refinement is a binary-relation that defines one *type* of transition. First, given the $\leq_t$ partial order, we say that one belief is more *concrete* than another if it is closer to either T or F. (Figure 3.3 illustrates graphically the concreteness relation in the basic bilattice.) Informally, if two attitude-beliefs are concrete, then they are equally refined. If neither is concrete, then one is more refined than the other if that deficiency is restricted to less relevant ($\leq_r$) propositions. Formally,

Definition 3.5.3 *Attitude-Belief Refinement*

Let $\langle A_1, B_1\rangle$ and $\langle A_2, B_2\rangle$ be attitude-belief assignments and let h be a proposition. $\langle A_2, B_2\rangle$ is an *immediate refinement w.r.t.* h of $\langle A_1, B_1\rangle$ (denoted $\langle A_2, B_2\rangle \sqsubset_h \langle A_1, B_1\rangle$) iff

1. for all $h' \in \mathrm{H} - \{h\}$, $\langle A_2, B_2\rangle$ coincides with $\langle A_1, B_1\rangle$; and
2. for h, either of the following holds:
   (a) $\mathrm{B}_2(h) \geq_c \mathrm{B}_1(h)$; or
   (b) $\mathrm{B}_1(h), \mathrm{B}_2(h) \in \{\mathrm{U}, \bot\}$, and $\mathrm{A}_2(h) \geq_r \mathrm{A}_1(h)$.

Note that $\sqsubset_h$ is a partial order (transitive and anti-symmetric). Let $\sqsubset \stackrel{\text{def}}{=} \cup_{h \in H} \sqsubset_h$ then $\langle A_2, B_2\rangle$ is an *immediate refinement* of $\langle A_1, B_1\rangle$ (without referring to a particular proposition) iff $\langle A_2, B_2\rangle \sqsubset \langle A_1, B_1\rangle$. Let $\sqsubset^*$ denote the transitive closure of $\sqsubset$. $\langle A_2, B_2\rangle$ is a *refinement* of $\langle A_1, B_1\rangle$ iff $\langle A_2, B_2\rangle \sqsubset^* \langle A_1, B_1\rangle$.

While useful for better characterization, refinement is often costly too. Thus, according to the GDD principle, *active* refinement should be confined to issues that are likely to affect repair. More generally, one wants to control any wasteful activity. For example, one may want to avoid repair activity that will not make a worthwhile difference to the patient state[6].

Definition 3.5.4 *Other Transition-types*

Let $\Gamma \stackrel{\text{def}}{=} \langle \phi_A, \phi_B \rangle$ be a state, h a proposition, and v an attitude or belief value.

1. An *inference* transition in the AB-space is one that is licensed by simple inference:

   - $\text{Infer}_{RB}(\Gamma,\ h,\ v)$ is the transition from Γ that is licensed by a one-step inference;
   - $\text{Infer-All}_{RB}(\Gamma)$ denotes an exhaustive sequence of inferences that starts at Γ (using an inference procedure such as 3.3.12).

2. An *action* transition-sequence in the AB-space is a sequence of transitions that is licensed by an action (or lack thereof).

   $\text{Act}_a(\Gamma)$ is a transition-sequence resulting from new information acquired by performing the action a in the state Γ. More generally, such transitions can be represented stochastically with several possible outcomes.

3. An *action-and-inference* transition-sequence in the AB-space is a sequence of transitions that correspond to an action (or lack thereof) and to subsequent exhaustive inferences.

[6] Before proceeding, I should concede that non-active refinement may also require significant resources and may thus have to be controlled too. Inference, for example, may require substantial computational resources, and time. While it may seem that I am ignoring computational complexity issues altogether, it is important to note that computation can also be *represented* as activity, if need be considered as such. I distinguish activity not so much because of what it requires from the agent, but more because of its potential effect on the patient. Interestingly enough, in trauma management, one does not even want to control all activity: *bed-side questions*, for example, are conveniently regarded as free and riskless.

Perform$_{a,RB}(\Gamma)$ is a transition-sequence resulting from new information acquired or inferred as a result of an action. In particular, Perform$_{a,RB}(\Gamma) \stackrel{\text{def}}{=}$ Act$_a(\Gamma)$; Infer-All$_{RB}(\Gamma')$, where Γ' is the state resulting from Act$_a(\Gamma)$.

For the purpose of controlling activity, it suffices to attend to an abstraction of the AB-space according to the Perform relation. That is, all transitions that are not in that relation can be ignored, and likewise all states that are not linked to any otherwise legitimate state by it. Given a specific attitude-belief describing the current state, we can also further constrain ourselves to states that are reachable from that state. The result is a situation calculus [101] in which situations are described as an attitude-belief assignment to a set of propositions, and in which Perform serves as an accessibility relation.

Given this situation-calculus and a current state, the problem is to decide on a sequence of action(-and-inference) transitions. Note however that, in this level of generality, this is precisely the basic planning problem with which we started, and which is unrealistically hard. What is new is that each of the state descriptions includes information about the relative relevance of acquiring certain knowledge and achieving certain conditions (goals).

In the ECM architecture, via the explicit encoding of goals, I hope to be able to avoid some of the complexity of that search. In particular, as further elaborated in Chapter 5, activity is controlled via three mechanisms in two levels. At the *goal level*, we have *goal setting rules* to distinguish *relevant* action transitions from irrelevant ones; and *goal inhibition rules* (cf. Section 3.6) to inhibit relevant but non-contributing or non-essential transitions. In the action level, the planner (cf. Chapter 4) uses goals to index sets of alternative procedures, and uses choice- and prioritization principles to select and order action transitions for a *combination* of goals. All three mechanisms, and combinations thereof, are rendered useful in important diagnosis-and-repair strategies catalogued in Chapter 7.

3.6 Goal Inhibition

One dimension along which strategies can be categorized is whether they are constructive or eliminative. While strategies are often designed to provide constructive advice, the on-going interplay between diagnosis and therapy in trauma management may sometimes require that pursuit of certain goals be delayed or even inhibited. While goal interaction can naturally be addressed by a planner, the planning task is notoriously complex, and it may thus be advantageous if unnecessary or non-contributing goals can be inhibited *before* they are passed on to the planner.

The basic approach to goal inhibition is to qualify goal-setting rules by the negation of any inhibition condition. Consider, for example a patient with a right lumbar wound. Normally, such injury would suggest a possible duodenal injury:

$$Right\_lumbar\_wound \triangleright Duodenal\_injury$$

A standard test for duodenal injury is a CT scan. However, this lengthy and costly procedure should *not* be pursued once there is a perceived need for a laparotomy since the surgical procedure will expose the duodenum anyway. Thus, one could qualify the above rule as follows

$$Right\_lumbar\_wound \wedge$$
$$unless(Laparotomy\_required) \triangleright Duodenal\_injury$$

While correct, the problem with such inhibition scheme is that if the inhibition condition is complex, rules become even more complex and hard to maintain. In addition, overloading rules makes them less interpretable: the separate function of each of a rule's antecedents becomes unclear (cf. [21]). Instead, inhibiting relationships between goals and between goals and conclusions can be *specified* separately. For each goal g, we can define an inhibition clause which will then be compiled (as a macro expansion) into all of that goal's goal-setting rules.

Definition 3.6.1 *Goal Inhibition Clause*

Given a goal g, *inhibit(g)* specifies the condition(s) under which g has to be inhibited.

Technically, the macro expansion procedure for a goal inhibition clause creates a new internal proposition called $inhibit(g)$ and a set of rules headed by that proposition that correspond to the condition itself. In the above example, the following new rule is added:

$$Laparotomy\_required \Rightarrow inhibit(Duodenal\_injury)$$

Then, every goal-setting rule for g will be padded as follows: Let $body(R)$ be the original body of a goal-setting rule R, replace $body(R)$ with

$$body'(R) \stackrel{\text{def}}{=} body(R) \wedge \neg true(inhibit(g))$$

An alternative, more elegant, approach to goal inhibition is based on the extension of GDD proposed in Section 3.4. In the extended framework, a goal-setting rule with a negated header can be thought of as representing an argument *against* pursuing that goal; exactly what we mean by inhibition. Taking that view, a goal inhibition rule is considered together with all reasons *for* its pursuit. Under this approach, goal inhibition rules are constructed as follows:

Let g be a goal, and let $inhibit(g)$ be an inhibition clause for g. Add a *goal-inhibition* rule:

$$inhibit(g) \mid\!\rhd \neg g$$

Under the inference procedure of Section 3.4, a *contradictory* attitude ($\perp$) will be assigned to a goal for which both a goal-setting and a goal-inhibition rule have succeeded. In our previous example, we will write:

$$Laparotomy\_required \mid\!\rhd \neg Duodenal\_injury$$

A third approach to *effectively* inhibit goals which is sometimes preferable in implementing diagnostic strategies is to "force" a concrete value (i.e. T or F) on the goal's underlying proposition. Such an approach is likely to be preferable when the belief in the inhibited goal *can* be inferred from the inhibiting condition; scaled diagnosis, discussed in the next example, is a case in point. Otherwise, this approach may not be elegant. For example, in the previous example, we could not conclude whether or not the patient suffers a duodenal injury. The net effect of that approach is that the goal remains relevant, but is regarded achieved by the planner and thus not actively pursued.

3.7 Example

In this section, I present a simplified version of TraumAID 2.0's diagnosis-and-repair strategy for *tension pneumothorax* (TP). The rules presented here were transformed from TraumAID 2.0's original knowledge base for the purpose of testing an implementation of the MVL-based formalization of GDD presented in this chapter. Rules are numbered, and appear in the order in which they were designed. The numbers, however, serve no purpose other than referential.

Diagnosis of TP follows a general strategy which I call *scaled diagnosis*, and which is further discussed in Chapter 7. Briefly, in this strategy, diagnostic activity proceeds cautiously by first acquiring information of low cost. Figure 3.4 presents the first two stages of the diagnostic part of this strategy. It starts with a report of a chest wound. Rule 400 initiates the first diagnostic (knowledge) goal, `Possibility_of_Tension_Pneumothorax` (PTP), by labeling its attitude R. When this goal is referred to the planner, and maybe depending on other needs, the planner may ask whether the patient is in shock and/or whether the patient suffers from distended neck veins. (The fact that each of these findings is an alternative means of satisfying this goal is represented in the planner's knowledge base.) If

```
; A chest wound initiates the process by setting first knowledge goal

400  Chest_Wound (Side . S)
        [> Possibility_of_Tension_Pneumothorax (Side . S)

; Initial physical evidence is gathered and used to continue or rule out

401  Chest_Wound (Side . S)
     Shock
        => Possibility_of_Tension_Pneumothorax (Side . S)

402  Chest_Wound (Side . S)
     Distended_Neck_Veins
        => Possibility_of_Tension_Pneumothorax (Side . S)

; In the case of positive evidence, the diagnostic process gears up

500  Possibility_Of_Tension_Pneumothorax (Side . S)
        [> Likely_Tension_Pneumothorax (Side . S)

; ...and more examination takes place

501  Possibility_Of_Tension_Pneumothorax (Side . S)
     Decreased_Breath_Sounds (Side . S)
        => Likely_Tension_Pneumothorax (Side . S)

; enough clinical evidence to justify testing

600  Likely_Tension_Pneumothorax (Side . S)
        [> Tension_Pneumothorax (Side . S)
```

Figure 3.4: Diagnosis of Tension Pneumothorax – Part 1

positive, any of these findings will confirm that there is a PTP, and the next knowledge goal, `Likely_Tension_Pneumothorax` (LTP), will be instantiated. Note that if it was immediately volunteered that the patient was in shock, then reasoning could immediately "leap" to that latter stage. If that was the case, then PTP's attitude would still have been labeled R, but it would not have been pursued since its belief would have been concrete (T in this case). To pursue LTP, again depending on other concurrent needs, the patient breath sounds would be assessed. The final diagnostic goal, labeled accordingly `Tension_Pneumothorax` (TP), is instantiated if LTP is confirmed.

```
; Alternatives: TP can be diagnosed via either X-ray or needle.

601  Tension_Pneumothorax_On_X-Ray (Side . S)
       => Tension_Pneumothorax (Side . S)

602  Needle_Aspiration_Chest_For_Pressure (Side . S)
       => Tension_Pneumothorax (Side . S)

603  not Tension_Pneumothorax_On_X-Ray (Side . S)
       => not Tension_Pneumothorax (Side . S)

604  not Needle_Aspiration_Chest_For_Pressure (Side . S)
       => not Tension_Pneumothorax (Side . S)

; Recommendation part. If unsure, treat "as if" positive.

700  Tension_Pneumothorax (Side . S)
       [> Rx_Tension_Pneumothorax (Side . S)

701  conflicted Tension_Pneumothorax (Side . S)
       [> Rx_Tension_Pneumothorax (Side . S)
```

Figure 3.5: Diagnosis of Tension Pneumothorax – Part 2

Figure 3.5 presents the final diagnostic stage, which was already discussed in Section 3.4 in the context of the reasoning about potentially contradictory evidence. Recall also that the therapeutic goal `Rx_Tension_Pneumothorax` (RTP) was recommended both when TP was confirmed, and also when there was contradictory evidence with respect to its presence.

Figure 3.6 describes the therapeutic part of that strategy. Notice that therapeutic goals are only labeled satisfied when there is *actual* evidence to that effect. Also note that there are two therapeutic goals, depending on whether the patient is stable or not. Finally notice the interaction of the therapeutic sub-strategy with the diagnostic sub-strategy in cases where a needle is used in the latter: a patient that was diagnosed through a needle aspiration does not need further decompression.

Goal inhibition knowledge is encoded in Figure 3.7 following the scheme by which lower-level goals are being subsumed by their successors when the latter are being concluded directly.

```
702  Chest_Tube (Side . S)
     unless Persistent_Tension_Pneumothorax (Side . S)
        => Rx_Tension_Pneumothorax (Side . S)

; If in shock, relieve it before proceeding with definitive treatment.

800  relevant Rx_Tension_Pneumothorax (Side . S)
     Shock
        [> Decompressed_Tension_Pneumothorax (Side . S)

; if a needle was used for diagnosis then already decompressed.

801  Needle_Aspiration_Chest_For_Pressure (Side . S)
        => Decompressed_Tension_Pneumothorax (Side . S)

; that's not a good way to do it, but will achieve the goal.

802  Chest_Tube (Side . S)
        => Decompressed_Tension_Pneumothorax (Side . S)
```

Figure 3.6: Treatment of Tension Pneumothorax

```
10400 Likely_Tension_Pneumothorax (Side . S)
         => Possibility_Of_Tension_Pneumothorax (Side . S)

10401 relevant Likely_Tension_Pneumothorax (Side . S)
         => Possibility_Of_Tension_Pneumothorax (Side . S)

10500 Tension_Pneumothorax (Side . S)
         => Likely_Tension_Pneumothorax (Side . S)

10501 relevant Tension_Pneumothorax (Side . S)
         => Likely_Tension_Pneumothorax (Side . S)
```

Figure 3.7: Tension Pneumothorax: Hierarchical Organization of Goals

3.8 Reasoning with Default Rules

Defaults have been shown to be useful in non-monotonic reasoning [130], and Ginsberg presents a corresponding *default bilattice* for his MVL framework (Figure 3.8). An advantage of using MVL as an underlying framework is the ability to replace the 4-valued bilattice with the default bilattice as the domain for belief and attitude.

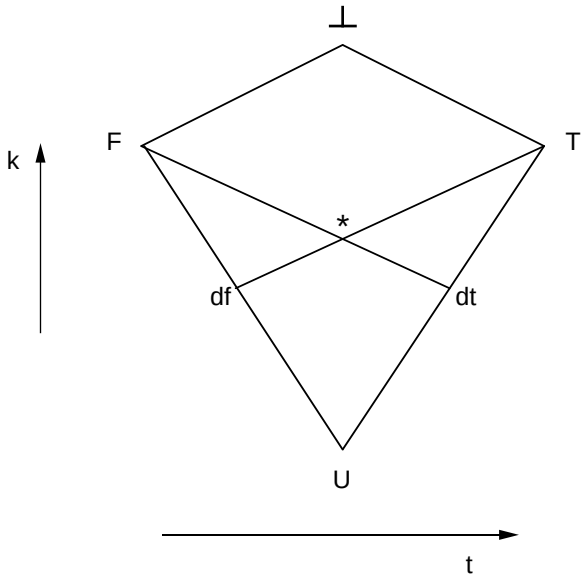


Figure 3.8: Basic Default Bilattice

In the default bilattice, *dt* and *df* represent a degree of truthfulness-knowledge that corresponds, respectively, to positive and negative default conclusions. With respect to the $\leq_t$ partial order, *dt* (and respectively *df*) is in-between the absolute unknown U, and the absolute truth, T (respectively F). Likewise, with respect to the $\leq_k$ partial order, both *dt* and *df* are inferior to either T, or F but represent more knowledge than U. Another new point is introduced, *, corresponding to a contradiction of default rules.

Defaults bilattices can also be extended to represent a *number* of *prioritized* defaults (i.e. which would one rather believe first, which second, etc.) Linearly ordered by the $\leq_t$ partial order, different default values can also be viewed as discretized

probability measures and may be useful in implementing diagnostic strategies in which a determinate conclusion cannot be made and a decision has to be made based on the most likely diagnosis. Figure 3.9 depicts a prioritized default bilattice.

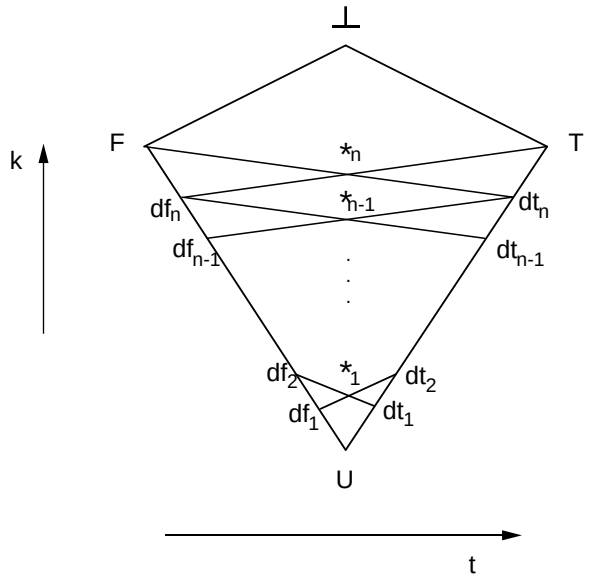


Figure 3.9: Prioritized (Ordered) Default Bilattice

Any number of positive/negative defaults can be defined. In fact, one can even have a *continuum* of positive and negative default points. Consider a mapping of the truthfulness×knowledge values of positive default points into the [0,1]×[0,1] interval and the negative default points into the [-1,1]×[0,1] interval, such that the point (0,0) denotes the bilattice U point, (1,1) denotes the T point, and (-1,1) denotes the F point (Figure 3.10). It is only important that monotonicity in both $\leq_t$ and $\leq_k$ is kept along both lines. An additional line of "contradictory" points extends up from the U point, taking values (0,k) for $0 \leq k \leq 1$, with the $\perp$ point denoted by the point (0,1). Points in this bilattice are combined according to a $\leq_k$ and $\leq_t$ partial orders which are usually defined by the truthfulness/knowledge coordinates. The only exception in the definition of $\leq_k$ is that in case of equal "knowledge" coordinates, points along the contradictory line are defined to be superior. Thus, use of the + operator to combine two points with equal knowledge value k but distinct truthfulness values yields the a "contradictory" value $(0, k)$.

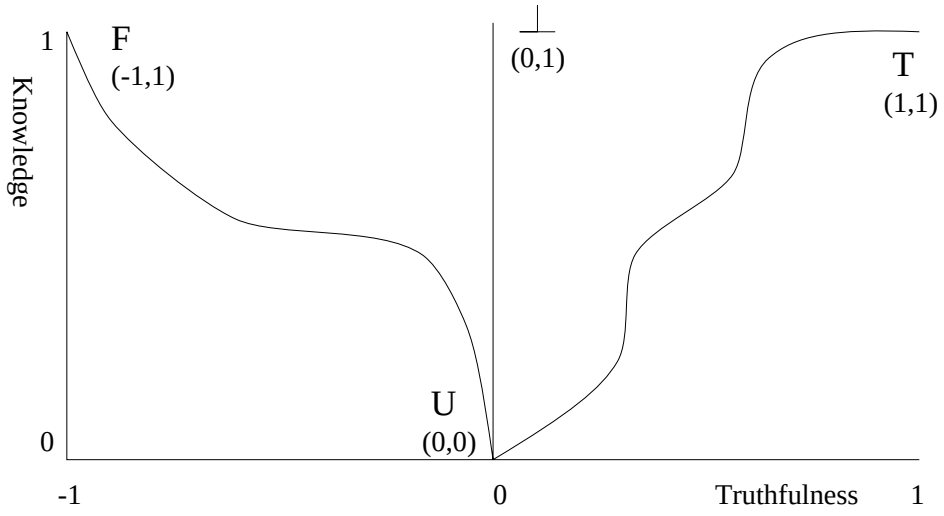


Figure 3.10: Bilattice with Continuous Defaults

GDD uses two bilattices, as domains for belief and attitude. However, as noted in Section 3.3, in extending a system's representation to a default bilattice, it is *not* necessary that both bilattices be the same. When the two bilattices are different, either the *relevant* functor or the *attitude-to-belief* function may have to be interpreted as a non-covering homomorphism from one bilattice to the other.

Of course, having a default bilattice as an underlying domain for belief and attitude is only useful if default rules can be written and used to manipulate such assignments. At present, TraumAID does not use any default rules (at least not explicitly). Next, we adopt Reiter's convention [130] by which a default rule has the form:

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\omega}$$

where α, β_i, ω are simply formulae.

The model-based semantic interpretation of this rule is as follows: if α holds, and if β_i's are all consistent, then one should default to models in which ω holds. More restrictive forms of default rules are *normal* (of the form $\frac{\alpha : \omega}{\omega}$) and *supernormal* (of the form $\frac{: \omega}{\omega}$).

Given a default rule, specified as above, it can be automatically translated (preserving its semantic interpretation) into a corresponding GDD evidential-rule[7] as follows.

Algorithm 3.8.1 *Representing a Reiter-style Default Rule*

Let R $\stackrel{\text{def}}{=}$ $\frac{\alpha_1,\ldots,\alpha_n:\beta_1,\ldots,\beta_m}{\omega}$ be a default evidential rule, where ω, α_i, β_i are all propositions, and α_i are interpreted as a conjunction. Translate R into the following GDD evidential rule.

$$\text{R'} \stackrel{\text{def}}{=} true(\alpha_1) \wedge \ldots \wedge true(\alpha_n) \wedge \neg false(\beta_1) \wedge \ldots \wedge \neg false(\beta_m) \wedge dt \Rightarrow \omega$$

For negative defaults, we do the same, only with $\neg\omega$ in the header[8].

Inference remains as before. Thus, default rules are merely another source of information, only one with a weaker confidence (which in Ginsberg's formulation, is represented by lesser knowledge capacity). Note that by that inference, if p is a proposition for which we have defaulted to dt, then if p is proved (respectively refuted) by another non-default rule, then p will be assigned T (respectively F). If, on the other hand, there is another applicable default rule for $\neg p$, then * will be assigned to p.

Having introduced an intermediate truth capacity, the goal interpretation issue arises again, i.e. should a proposition for which a *dt relevance* was assigned be adopted as a goal? For adopted goals, what is an acceptable degree of satisfaction with their achievement: is it enough to conclude dt for a goal proposition? This is, of course, highly subjective and should thus be left to implementation.

[7]Similar procedure applies for goal-setting rules.

[8]Note that this is different from using *df* as the rightmost antecedent of a rule with ω in its header.

3.9 Goal-Directed Reasoning in TraumAID 2.0

TraumAID 2.0's diagnostic reasoner embodies most of the ideas of the GDD paradigm just described, but not its formal details. The latter is, in fact, a later formalization of the *kind* of reasoning used by TraumAID 2.0. Next, for the sake of completeness, I will describe the details of this implementation. This description should also convince the reader that the GDD formalization captures and distills the principles and spirit of TraumAID 2.0's diagnostic reasoning, although presents them in a more general and elegant way. The MVL-based formalization of GDD was implemented separately.

Built on top of TraumAID 1.0's inference engine, TraumAID 2.0's diagnostic reasoner uses *facts* to represent primitive propositions[9]. The first difference between the implementation and the formalization is that each such fact is assigned a *single* truth value (roughly speaking, representing belief), compared with an attitude and belief in the formalization. In addition each such truth value is restricted to the set { T, F, U }. Thus, TraumAID 2.0 does not have the richness of conflicts and defaults.

Here too, rules are used to represent knowledge. As in Section 3.3, rules are only allowed to have *positive* headers. A rule may succeed, if all its antecedents are *known* to hold (i.e. T, or F), fail, if one of its antecedents is *known* not to hold (F), or be inapplicable if one of its antecedents is unknown. Inference follows a forward chaining paradigm[10] where a proposition is assigned a truth value T if *one* of the rules for which it serves as a header succeeds. It is assigned a truth value F, if *all* such rules fail. Otherwise, if some but not all of such rules fail (others being

[9] Facts may have attributes associated with them. Although those attributes may be existentially quantified, we can safely regard facts as primitive propositions since the domains for attributes are finite, and often small.

[10] While for the most part inference follows the forward chaining paradigm, TraumAID 2.0 still has some *leftover* backward chainers. In TraumAID 1.0, backward chaining was used for information acquisition. While in TraumAID 2.0, planning is used for that purpose, some rules which are backward chained to acquire bed-side questions were left. Imposing a planned information acquisition process on those is possible and was not done solely as a matter of convenience.

inapplicable), it is assigned a U value.

TraumAID 2.0 has rules of a single type, which are used to implement both goal-setting and evidential reasoning. The semantic distinction is made by separating two types of facts: *conclusions* and *goals*. Different facts represent a condition (e.g. *Pericardial_Tamponade*), and goals associated with that condition (for example, *RO_Pericardial_Tamponade* the goal of learning about that condition, and *Rx_Pericardial_Tamponade*, the goal of treating it once it was diagnosed). Since conclusions and goals are represented in different facts, additional machinery is used to link them (for example, in order to avoid the pursuit of the diagnostic goal when one already has knowledge about the disease by other means).

Consider again the diagnosis and treatment of a pericardial tamponade (Example 3.3.6). The corresponding rules in TraumAID 2.0 are:

1. **Rules for setting a diagnostic (knowledge) goal:**

   $(\ldots) \rightarrow$ *RO_Pericardial_Tamponade*

   "It is relevant to *know* if the condition holds".

2. **Rules for satisfying a diagnostic (knowledge) goal:**

   $(\ldots) \rightarrow$ *Pericardial_Tamponade*

   "Conclude whether the condition holds".

3. **Rules for setting a therapeutic goal:**

   $(\ldots) \rightarrow$ *Relieve_Pressure_Pericardial_Sac*

   "It is relevant to *address* the condition".

4. **Rules for satisfying a therapeutic goal:**

   $(\ldots) \rightarrow$ *Successfully_Relieved_Pressure_Pericardial_Sac*

   "Conclude whether the condition has been successfully addressed".

TraumAID 2.0 has another mechanism, called *goal-hierarchy*, made of goal inhibition clauses which are automatically compiled into the rules. Initially, inhibition conditions were limited to a description of a partial order (hierarchy) among goals. For a given goal, these indicated at the presence of which other goals it should be inhibited. These inhibition clauses took the form:

(inhibited-goal (inhibitive-goal-1 $\cdots$ inhibitive-goal-n))

where the inhibited goal is to be inhibited whenever *any* of the inhibitive goals is determined to be relevant.

For example, this hierarchy could be used to represent a goal-specialization relationship by inhibiting a general goal at the presence of any of its specializations, as in

.

```
(RO_Possibility_Of_Esophageal_Injury)
  ((RO_Lower_Thoracic_Esophageal_Injury)
   (RO_Upper_Thoracic_Esophageal_Injury)))
```

Since it may sometimes be useful to inhibit a goal when a certain *fact* holds, the syntax of an inhibition clause was then extended to allow any fact in the antecedent part.

To summarize, TraumAID 2.0's embodies an implementation of GDD that is similar to, albeit not the same as, the MVL-based formalization presented in this chapter. In the formalization, which chronologically followed the implementation, I tried to create a version of GDD that is cleaner and more elegant than the original implementation, while preserving the original flavor.

Chapter 4

An Approach to ECM Planning

4.1 Progressive Horizon Planning

4.1.1 Planning as Search in a Situation-Action Space

Classical planners originated from theorem-provers and search-based problem solvers such as GPS [63, 111, 43]. There are several ways in which planning can be viewed as a search task: Classical planners (e.g. Strips [44], Noah [137], Sipe [159], Hacker [144], Nonlin [150]) view planning as a search in a space of plans. In a plan-space, states represent plan structures and transitions correspond to transformations that take one plan structure and transform it into another (e.g. add or remove action, impose or change order, refine the detail, etc.[1]) Alternatively, planning can also be viewed as search in a space of world state descriptions, connected with edges representing world state transformations (e.g. [101, 88, 59, 161]).

While in the context of planning action-based transformations are clearly of special interest, other operators may also be relevant. Consider, for example, the Attitude-Belief (AB) space of Chapter 3. It qualifies as such space as its states can be viewed

[1] As aside, there is a common distinction in the planning community between those planners who work in the space of *partially* ordered plans, and those that consider linearized plans (see [106] for a relevant discussion).

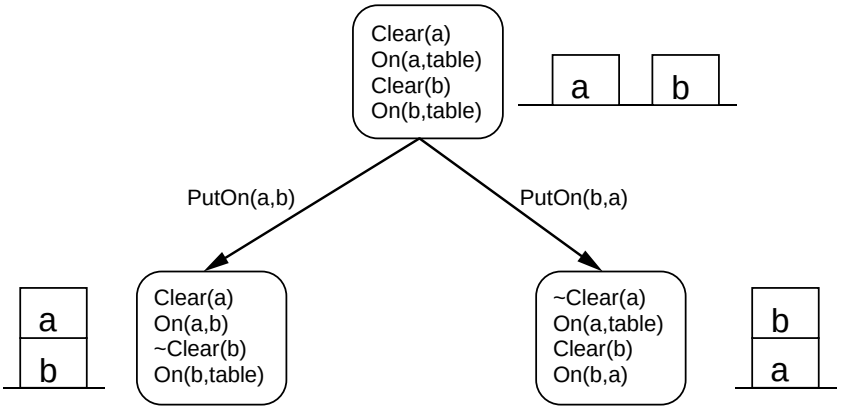


Figure 4.1: Situation-Action Tree.

as subjective (and also introspective) world descriptions. In addition to action-based transformations, it also has transformations based on internal inference, volunteered information or unexpected events, etc.

By a *Situation-Action* (SA) space, I mean a space of world descriptions as above, which is abstracted to only reflect agent-controlled action-based transitions (including the special *no-action* transition). Assuming for a moment that world descriptions are complete, and yet manageable, and ignoring the effects of uncertainty and unpredictability, a planning task can be defined as follows:

> Given an initial state, a repertoire of actions, and a goal-state description, find a sequence of actions (path in the SA-space) that transforms the initial state to a goal state.

Viewed that way, planning can be accomplished via simple search for a goal state in a (possibly infinite) directed tree[2], rooted at the initial state. I call this the SA-tree. Figure 4.1 depicts a simple SA-tree.

Obviously, planning by simply searching an SA-space representation is not a practical enterprise. However, since it so closely resembles a *simulation* of reality, plans

[2]I simplify here as this may actually be a graph, and may even contain cycles, i.e. sequences of actions that "cancel out" one another. However, I will only use the SA-tree as a presentation metaphor, and so this simplification does not represent a problem.

constructed using other representations may easily be transformed into the SA-space, making it attractive as a common ground for evaluating artificial (and also human) planning techniques.

One way in which the SA-space resembles a simulation of reality lies in how it can capture *uncertainty* and *unpredictability*. As a matter of terminology, by uncertainty I refer to lack of knowledge which can be remedied via exploration, whereas by unpredictability I refer to unexpected events or outcomes that can only be determined *after the fact*. Alternatively, of course, both uncertainty and unpredicatability can be dealt with by making plans that are more robust and/or by conditionalizing one's plans to accommodate multiple outcomes. In the SA-space, unpredictability means that a given action performed at a given state may result in one of a *number* of possible states, each corresponding to a different outcome. Search-wise, unpredictability results in an increased *branching factor*. Uncertainty means that one may not *know* what exact state one is in, so one may have to expand the plan to include more diagnostic activity.

Consider the effect of uncertainty on the previous example, as depicted in Figure 4.2. If the agent does not *know* whether a is a block or a pyramid, it has one of two options: either to go ahead with a plan that does *not* verify a's nature, or expand the plan with an exploratory move. Note, however, that if not for the uncertainty, exploratory actions could have been eliminated without any adverse effect on the plan's correctness. Uncertainty thus contributes to the *length* of the plan. Unfortunately, in a simple brute force search, the plan's length is exponentially related to the size of the search space, and consequently its run-time.

The Perform-abstracted AB-space (cf. Section 3.5) is an example of such SA-space.

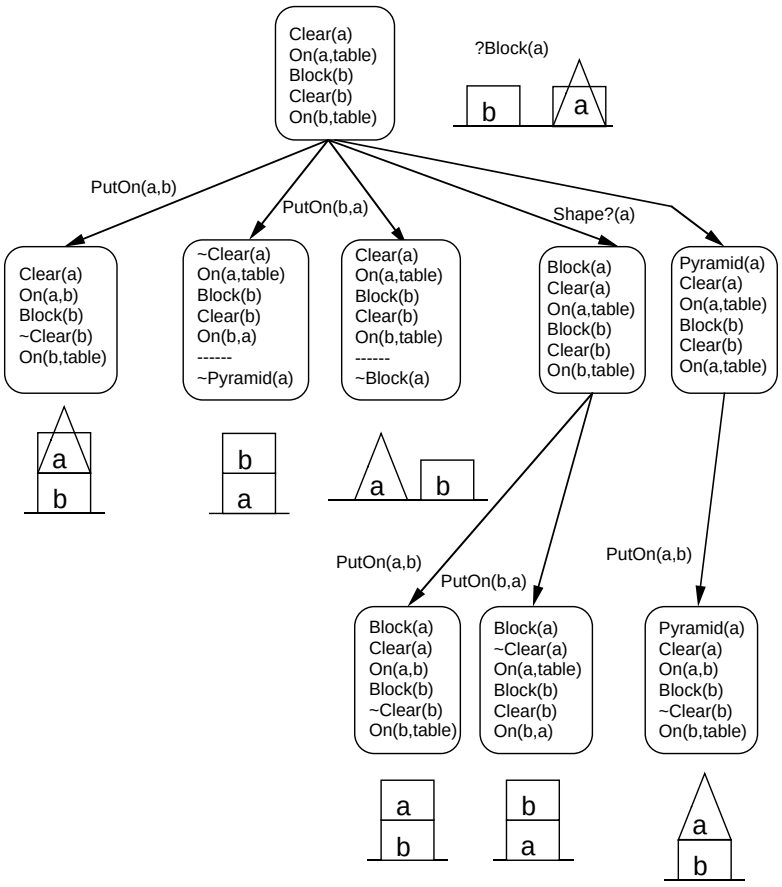


Figure 4.2: SA-tree with Uncertainty

4.1.2 Complete Plans that are Optimized to a Horizon

While the SA-space-based characterization of the planning task is methodologically useful, it is hardly useful in an algorithmic sense. Given the complexity of planning as search, in both the plans-space and the situation-action space [14], much of the research in that area has focused on ways to reduce search. Some common tactics includes:

- Use of heuristics in focusing the search, e.g. [66, 43];
- Resource-bounded search, e.g. [89, 91, 134];
- Means-ends analysis, e.g. [44];
- Search for partially-ordered plans, e.g. [137, 150, 160];

- Least commitment approach, e.g. [142, 74];
- Abstraction of plan and operator descriptions, e.g. [136, 85, 152, 46];
- Defaults and approximate planning, e.g. [41, 62].
- Search-space decomposition, e.g. [94, 109, 157], operator decomposition, e.g. [86], and competence level decomposition, e.g. [10].
- Reactive planning via off-line computation, e.g. [47, 55, 138], or use of deictic representations, e.g. [6].
- Off-line and on-line learning, e.g. [45, 65, 105]

Many of these approaches are *partial* in one way or another. For example, the search performed by a planner may be incomplete, it may produce parts of plans only (e.g. next action models), or it may rely on unproven or default assumptions.

Progressive Horizon Planning (PHP) is an incremental planning framework. It works within a cycling architecture, such as the ECM architecture, in which planning is strongly coupled to a characterization-oriented reasoner and to an action/perception component. In PHP, planning is an on-going task: the eventual plan (*The Plan*) is being shaped while it is executed. In particular, intermediate plans, constructed in each cycle, are partially followed and then adapted based on the response and other events. Note that since part of the purpose of action in exploratory-corrective domains is to determine goals, it is unlikely that any of the intermediate plans will be complete. However, while each intermediate plan is thus partial in the sense that not all goals are known, in PHP *all known* goals are somehow addressed in every intermediate plan. PHP is thus *partial-global*.

I have already made the point that a plan must at least be reassessed whenever a new and relevant piece of information becomes available. Since in the ECM architecture this may happen in every cycle, intermediate plans must be re-constructed (or at

least adapted) many times throughout a management session. Given the computational complexity of planning, PHP planners focus their computational effort on an intermediate plan's *initial segment*. In the case of TraumAID 2.0, this approach is supported by characteristics of the multiple trauma management domain, and of the ECM architecture within which its PHP is embedded:

- First, note that "*The Plan*" — the sequence of actions that is *actually* followed by the agent – is intended to be the ordered concatenation of the initial segments of the individual intermediate plans[3]. Thus, we can be less sensitive to a given intermediate plan being less than optimal if its inadequacy can be corrected later.

- Second, with respect to dependencies between earlier and later actions, practice standards in the trauma management domain provide well-defined constraints on the relative urgency and importance of problems. In addition, the limited interaction between goals in the trauma domain (cf. Section 4.2) reduces the chance that the overall plan would be adversely affected by a premature decision in addressing a most urgent, most important, goal.

- Third, note that uncertainty and unpredictability mean that *any* plan, unless specifying all contingencies, may become obsolete during execution. Since unpredictability increases with the time between conception and execution, later parts of the plan are more prone to it. Similarly, since earlier parts of the plan may be aimed at removing uncertainty in support of later decisions, one expects less uncertainty closer to the time in which such decisions have to be made. Thus, focusing on a plan's initial segment is sensible. In a way, we are taking a computational advantage of the uncertainty-unpredictability disadvantage.

The term *horizon* is often used to describe the depth of search, and is commonly determined by the available computational resources. Famous is Korf's *Real-Time*

[3]Intended, but not necessarily, since the agent can choose not to follow these plans.

A^*[89], a planning algorithm which is based on the Iterative Deepening A* search technique[4] and searches an action-based search space to a horizon. If a goal state is encountered during search, then actions along the path from the initial state to that state make up a plan. Otherwise, if a goal state has *not* been reached, the distance between the node at the horizon and a goal state is heuristically evaluated. The node which is thought to be closest to a goal is picked, and the actions along the path from the root to that node make up the chosen plan initiation. Given the above characteristics of the multiple trauma management domain and of the ECM architecture, RTA* is clearly a strong candidate for an ECM planner.

Recall however that we required that intermediate plans be *global*, i.e. that they will address *all* known goals. In the context of consultation, plan completeness is important to sustain a person's cooperation. In addition, as will be demonstrated next, *sketching* a complete solution can ensure a certain level of consideration for later parts of the plan.

As in RTA*, the first basic idea in PHP is to focus computational efforts on the plan's initial segment. The second basic idea is that it is often easy to construct an approximate plan, or a *plan sketch*. Plan sketching is strongly related to abstraction; specifically it represents an abstraction along some "goodness" metric (as opposed to the plan's completeness, or level of detail). In my analysis, I will assume that sketching of entire plans, or of plan completions, can be done efficiently. Plan sketching will allow us to consider and present complete plans. In Section 4.2, I describe an accompanying technique for plan sketching.

Suppose then that we have an efficient plan sketching algorithm. One approach to PHP is essentially a variant of RTA*:

[4]Iterative Deepening A* (IDA*) [87] is a general-purpose search technique which, via a number of depth-first passes, simulates A* but avoids its space requirements. More recent linear-space A*-like algorithms, e.g. [91, 134] can also be used in RTA*.

Algorithm 4.1.1 *RTA*-based PHP*

- Search as in RTA*; when evaluating a node (at the horizon) do the following:
  1. Sketch a completion to the plan initiation represented by the actions along the path from the root to that node;
  2. Evaluate the completed plan.

This view of PHP is advantageous primarily because it lends itself to complexity analysis: the *order* of complexity added to RTA* by the plan completion sketching represents an additional *factor*, since it has to be applied to every node at the horizon. Since we assume plan sketching to be cheap, the addition is minor compared to a complete search. Figure 4.3 illustrates the complexity graphically. It depicts an SA-tree rooted at the initial state; a plan which is a path in that tree leading from the initial state to a goal state; and the PHP horizon. Making some regularity assumptions on the tree's structure, let a be the tree's branching factor, let n be the plan's length, and let C_{opt} be a *constant* denoting the depth of the search horizon, then

1. Assuming it is done breadth-first, a complete run-time search requires $O(a^n)$ *time*. It is geometrically represented in the figure by the triangular area of the entire tree, from the initial state down to the depth of the plan;
2. Reactive planning techniques that pre-plan for all contingencies may require, for similar reasons, $O(a^n)$ *space* (and therefore so much *pre-processing* run time as well) [61];
3. In *RTA**, assuming a constant-time evaluation function, $O(a^{C_{opt}})$ time is required, represented geometrically by the area of the mildly shadowed triangle at the top of the SA-tree;

4. In the RTA*-based PHP, one needs to sketch a completion for each node. Assuming that plan completion can be done in time that is linear in the length of the plan, then this process takes $O(n \cdot a^{C_{opt}})$ time. In the figure, the added cost is represented by the area of the dark-shadowed rectangle spawned from the horizon down the SA-tree.

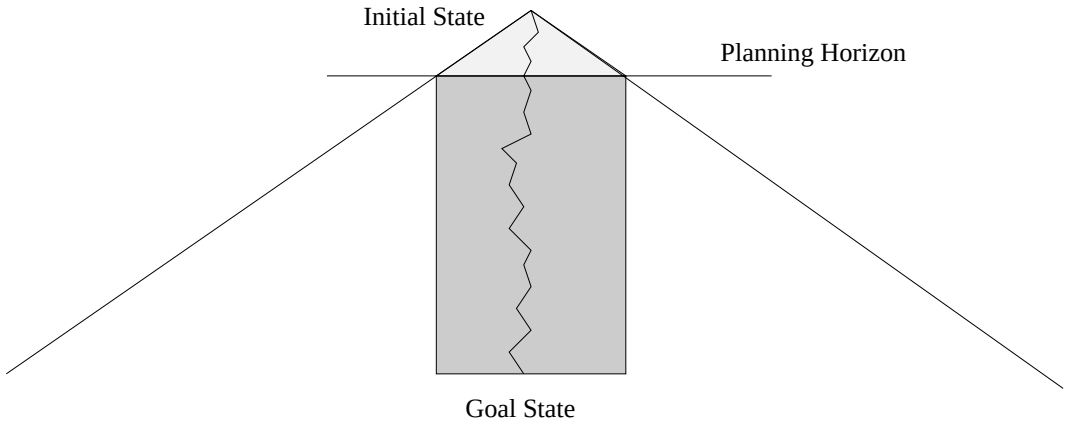


Figure 4.3: Complexity of PHP versus Complete Planning and RTA*

A second approach to PHP, which is also the one that is actually used in Traum-AID 2.0, involves a two stage construction:

Algorithm 4.1.2 *2-Stage PHP*

1. Sketch a complete plan;
2. Optimize that plan to a horizon;

   Repeat steps 1,2 if necessary.

In this approach, a complete plan sketch is first constructed, and then a number of horizon-restricted optimizations is applied to it. These optimizations, while applied to the plan's initial segment, can and should consider interaction of this segment with later parts of that plan sketch. Optimization routines can often be described as plan

transformations[5]. They can be either domain specific or general to the particular plan representation and/or sketching algorithm, but should importantly be *local*, i.e. a plan-space search is *not* expected. In Section 4.3, I will present examples of such optimizers taken from TraumAID 2.0's PHP algorithm.

This two-stage process may then have to be repeated. Specifically, sketch completion may have to be repeatedly applied if the post-optimization plan is incomplete, i.e. does not address all goals. The optimization step may have to be repeated in case where the post-optimization horizon changes and is now made of different actions.

The run-time of the 2-Stage PHP is harder to analyze than that of the RTA*-based variant without assuming a certain cap on the number of times the two-stage process is repeated. However, under the *very weak* assumption that it is not repeated more times than the overall number of possible plan initiations ($\leq a^{C_{opt}}$), and under the additional assumption that optimization and sketching are both linear, this time is well bounded below the RTA*-based variant's run time. More commonly, the 2-stage process will only be repeated a few times, resulting in a linear algorithm.

4.2 Planning as Means-Selection and Ordering

Most planners take planning as the construction of a sequence of actions, or of another representation of a similar function (e.g. conditional plans), to satisfy a *given* set of goals. In Section 4.1, I argued that this assumption does not hold in general in exploratory-corrective domains where goal characterization is part of the purpose of planning, and have therefore suggested the ECM/PHP framework. Nevertheless, within the ECM framework's plan sketching process, goals (some of which are indeed exploratory in nature) can be assumed given. In this Section, I focus on a particular formulation of an ECM plan sketching problem in which the planning task is *functionally* divided into two sub-tasks:

[5]Note that we essentially move to the plan-space now.

1. The *Selection* sub-task in which a set of procedures has to be chosen which parsimoniously (with respect to some pre-defined measure of preference and cost) address the current combination of goals; and

2. The *Ordering* sub-task in which these procedures have to be ordered with respect to one another to form a single overall plan, taking into account each procedure's urgency, priority, compatibility, etc.

Such functional separation is attractive in part because each of the sub-tasks can be formalized as a well-studied, abstract, *domain-independent* problem. Specifically, the selection sub-task can be formulated as a *set-covering* problem whereas the ordering sub-task can be formulated as a constraint-based *scheduling* problem. In what follows, I first review related work and identify some of the assumptions that are necessary to support this formulation. Then I present the formulation itself. Finally, given that selection and ordering are often co-dependent, I present a *selection-and-ordering* planning algorithm which *interleaves* the two.

4.2.1 Assumptions

In the planning literature, the term *conjunctive goal* is often used to describe a situation in which a plan has to satisfy *several* goals. Many planners, including most of the pioneering work as well as many later domain-independent planners (e.g. [44, 14, 100]), make few assumptions on the nature of sub-conjuncts and their interaction with one another.

In many of these planners, sub-conjuncts are taken simply as describing various *aspects* of the goal-state, and are treated equally. In some other frameworks, these description-parts are *categorized* and serve different purposes in the planning/execution algorithms. Schoppers' Universal Planning algorithm [138], for example, distinguishes between *sub-goals* and *qualifiers*.

While the effect of goal interaction is commonly acknowledged by planning researchers (for example "interactions between steps, [are] a common cause of bugs" [144]), few have studied interactions in more than a general form. Commonly, interactions are resolved by way of *search backtracking*, or via the application of *critics* after the plan is constructed. Exceptions include work by Drummond and Currie [40] exploring various goal-ordering schemas to be applied *while planning*, and work by Cheng and Irani [15] formalizing intra-operator sub-goal ordering and showing that some constraints can be inferred even before planning starts. In more recent work, Barret and Weld [7] have studied the effect of goal "serializability", i.e. the extent to which the order in which goals are attended affects the effort required to solve the problem, on various types of planners. Hayes' work on *Machinist* [67], a domain-dependent planner which *uses* interactions to guide its search, is another interesting exception. Hayes pinpoints the difficulty: "the ability to identify a goal interaction efficiently by looking at a problem specification requires intimate knowledge about that problem domain".

Nau *et al.* [109, 162] present a planning framework in which a *limited interaction* assumption is adopted, and show it results in a significant reduction in search. In my framework, consistent with the ECM architecture's decomposition of reasoning, I can assume that goal interaction is limited to competition for activity and other resources[6]. This allows a much simpler formulation and solution of the planning problem.

My first assumption is thus that multiple goals represent multiple objectives and not multiple aspects of the same goal state and that goals are independent except for the fact that they compete for the same resources, e.g. the agent's attention, time, etc. In particular, I assume there are no *clobberers/white knights* [14], nor *causal links* [100] between actions aimed at different goals.

My second assumption is that it is not hard to *a priori* construct plans to satisfy

[6] The framework will actually allow other forms of interaction as well.

individual goals; I call these *procedures* or *protocols*. These procedures can be computed on-line, as in [94, 109] for example, or simply pre-specified. I take planning to be the problem of *combining* individual plans into a single *efficient* plan (see also [48]).

Most importantly, my third assumption is that the domain of application can often indicate certain constraints and preferences. Of particular importance for our formulation are *choice* alternatives and preferences (among goals, among alternative procedures for an individual goal, among alternative combinations of procedures, etc.), and *ordering* preferences and constraints (among goals, among procedures for different goals, and between parts of procedures). Muscettola and Smith's HSTS [108], a system for planning telescope observations, also integrates planning and scheduling techniques; among other things, HSTS provides a *domain description language*.

The assumptions and intuitions of Nau *et al.*'s framework are very close to mine. Although different, their formalization and algorithm seem to make a suitable alternative to the ones I am about to propose. The forms of interactions considered by Nau *et al.* are:

1. *action-merging* interaction by which a set of actions A can be replaced by a *merged action* $m(A)$, which has a lesser cost. In my set-covering framework, such interaction is handled by seeking covers which are minimal with respect to some individually defined cost function;

2. *action-precedence* interaction by which a certain action a in one sub-plan P must precede another action a' in another plan P'. Both frameworks handle action precedence via the introduction of ordering constraints;

3. *identical-action* interaction occurs when one action appears in several subplans. This interaction is problematic because it may create contradictions (cycles) in the action-precedence graph. We will handle this sometimes by choosing alternative procedures or, when possible, by relaxing the action-precedence requirements;

4. *simultaneous-action* interaction, when different actions *must* occur at the same time. We will not have such interaction in our framework.

My framework allows other constraints between procedures and also between their respective *goals* that I believe are not supported by Nau *et al.*'s. For example, one may demand that an action aimed at an urgent goal *not* be preceded by a lengthy action (elapsed time as a resource; I treat time in a way that is similar to Vere's DEVISER [154]). Other examples include compatibility constraints, preferences, etc. In principle, though, Nau *et al.*'s framework can be easily extended to accommodate such constraints.

Both frameworks permit alternative plans for a given goal. In both frameworks such procedures can be ranked, which is useful for search prioritization. One difference, however, is that in my framework procedures are ranked by both by their preference for an individual goal and, *independently*, by their respective cost/risk. This is useful because it allows specifying a *partial* treatment as a less desirable, although often less costly, alternative. In such instance, my algorithm will opt for the partial treatment if none of the complete treatments is feasible given the interactions with other goals.

In both frameworks, an *efficient* plan is sought. This is possible because both frameworks assume that a goal can be satisfied via a number of alternative plans. In my framework, most of the efficiency is derived from the fact that a single procedure may often be used for several goals (although possibly not at the same level of preference for all). Nau *et al.*'s framework is more general: such efficiency is defined for combinations of procedures using the $m(A)$ operation. Finally, while

my set-covering formulation defines a whole space of cost-ranked solution-plans, the plan-sketching algorithm is limited to a greedy depth-first search in that space. Nau *et al.*'s algorithm can be viewed as a more complete best-first approach.

4.2.2 Means-Selection as Generalized Set Covering

Given a certain combination of goals, each of which can be addressed via a number of alternative procedures, a set of procedures is needed which addresses them all. In cases where multiple solutions exist, one may wish to maximize some preference criterion, defined over all such sets. For example, one that minimizes risk or cost, maximizes the likelihood of success, etc. Presented at this level of abstraction, this problem resembles many *abduction* problems, e.g. abductive explanation and diagnosis [129, 119, 11, 28, 97]. Set-Covering [78, 75, 16] is an abstract mathematical problem which was previously used to formalize abductive reasoning (e.g. [129, 4]), and which I next use to formalize the means-selection sub-task.

Definition 4.2.1 *Goal-Procedure Mapping*

Let GOALS $\stackrel{\text{def}}{=}$ $\{ \text{G}_i \}_{i=1}^{n}$ be a set of goals, PROCS $\stackrel{\text{def}}{=}$ $\{ \text{P}_j \}_{j=1}^{m}$ a set of procedures. A *goal-procedure mapping* is a function from GOALS to the power set of PROCS which indicates, for each goal, all the procedures that can alternatively be used to address it:

$$\text{addressable-by(g)} = \{p \mid p \text{ can address } g\}$$

Conversely, each procedure can be associated with the goals it "covers":

$$\text{addresses(p)} = \{g \mid p \in \text{addressable-by}(g)\ \}$$

Note that each goal is addressed *alternatively* by a number of procedures, whereas a procedure addresses *simultaneously* several goals.

Definition 4.2.2 *The Procedure Selection Problem*

An instance of a *Procedure Selection* (PS) problem includes a set of *goals*; a set of *procedures*; and a set of *goal-procedures* mappings as above. In addition, a *cost* function is individually specified for every procedure. Then, given a subset of goals G$\subseteq$GOALS, the problem is to find a subset of procedures P$\subseteq$PROCS that satisfies the following conditions:

1. *completeness criterion*: all goals in G are addressed by at least one procedure from P;

2. *cost optimality*: P has a minimal overall cost among all subsets of PROCS satisfying condition 1;

As stated, the PS problem is isomorphic to an optimization version of an NP-Complete problem called Set Covering.

Definition 4.2.3 *The Set-Covering Problem*

Let F be a collection of subsets of a finite set A, such that $\cup_{i=1}^{n}$ S$_i$=A, and let $B \subseteq A$. A cover of B in F is a subset $F' \subseteq F$ such that $\cup_{S_i \in F} S_i = B$. Given F, B, and an integer k, is there a set F' such that

1. F' is a cover of B in F;

2. $|F'| \leq k$.

The isomorphism is obtained by identifying GOALS with A, G with B, the set $\{\text{addresses(p)}\}_{p \in PROCS}$ with F, assuming uniform cost, and finally taking P to be the procedures corresponding to the subsets chosen for F'. Since this isomorphism can be computed in polynomial time, the PS problem is NP-Hard [53]. While as stated above, the PS problem is more general than the Set-Covering problem, optimization versions of Set-Covering do exist and use cost functions in a way that

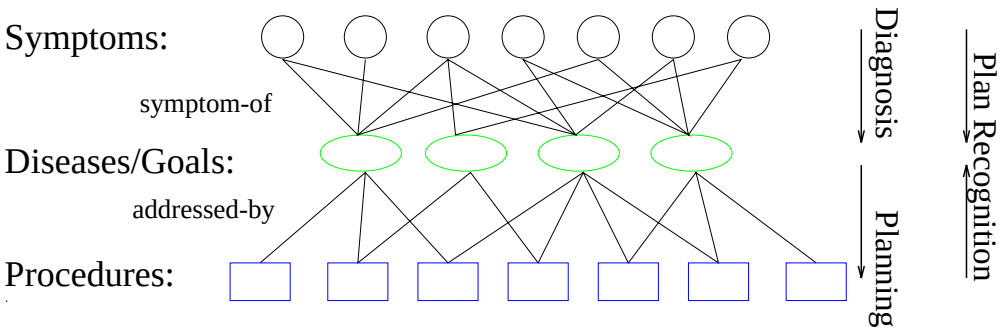


Figure 4.4: Diagnosis, Planning and Plan Recognition as a Set Covering Problem

is similar to their use in the PS problem. Interestingly, albeit the notorious NP-Hardness, Chvatal [16] shows that a greedy algorithm performs pretty well on a similar optimization version of Set-Covering.

Using Set Covering to formalize the selection sub-task is also interesting because of its use in formalizing the related tasks of *diagnosis* and *plan recognition*:

- Reggia *et al.* [129] formulate diagnosis as a set-covering problem in which a set of *diseases* has to be selected such that it "covers", or explains, a given set of observed symptoms. Diagnostic problems are represented as bi-partite graphs with diseases in one of its partitions, and symptoms in the other. Each disease is connected to all the symptoms it may cause.

  Thus it may be interesting to *conceptually* model both diagnosis and planning as a two-stage set-covering problem. Figure 4.4 illustrates a process in which a set of symptoms is mapped, via set-covering, to a set of possible/verified diseases (diagnostic/therapeutic goals), which are then mapped to a set of procedures.

- A plan recognition problem can be defined as follows: given a set of observations, including but not limited to symptoms and actions taken by another agent, try to hypothesize that agent's goals. Considering Figure 4.4 again, plan recognition can also be formalized as a set covering problem, this time from observed symptoms and procedures to goals. Such view is related to Kautz's

formulation of plan recognition [82], except that it represents a much simpler form of circumscription.

4.2.3 Ordering as Constraint-Based Scheduling

Consider a *given* set of procedures. Ordering them into a single coherent plan represents a classical *scheduling* problem. For the purpose of this section, I will take a plan to be a partially ordered set of procedures represented in a *plan graph* – a directed acyclic graph in which nodes correspond to procedures and arcs represent a precedence order. The formulation presented is simple but can be extended without much difficulty to support a richer notion of resources, stochastic durations of processes, richer representation for overlapping processes, etc.

Definition 4.2.4 *Precedence Constraints*

A precedence constraint is a pair (P_i,P_j), where P_i, and P_j are procedures, representing that P_i should precede P_j.

Some precedence constraints are fixed, e.g. a given procedure must *always* precede another, whereas others are dynamic and can only be determined by the context in which the two procedures are called for, e.g. by the respective urgency of the goals for which they were selected. Precedence relations are used to reflect interactions between the actions, their respective goals, resources, etc. One way in which precedence relations can be represented is in a plan-graph:

Definition 4.2.5 *A Plan-Graph*

Given a set of procedures PROCS and a set of precedence relations C, a plan-graph is the directed graph G(PROCS,E) in which each precedence relation (P_i,P_j) in C is represented in E as a directed arc from P_i to P_j in C.

Thus, a given set of procedures and a corresponding set of precedence constraints define a plan-graph. However, given a set of constraints, there might be *no* plan that meets all constraints. In other cases, there might be more than a single plan which satisfies all constraints.

Proposition 4.2.6 Given a set of procedures PROCS and a set of precedence constraints C, a valid plan exists iff the plan-graph G(PROCS,C) contains no cycles. Multiple valid plans exist if an arc can be added to the transitive closure of C without creating a cycle.

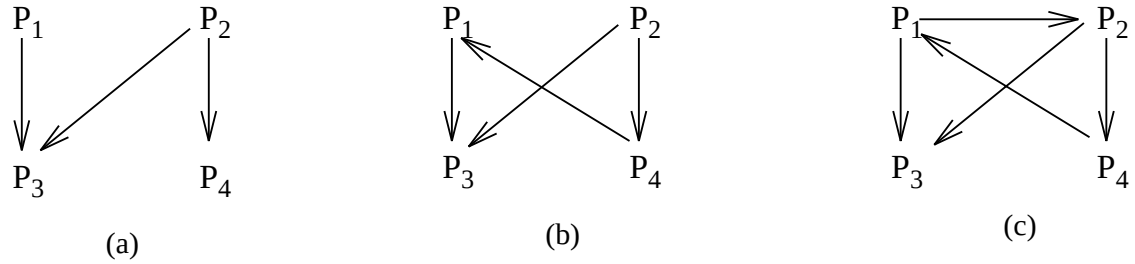


Figure 4.5: Plan Graphs

Example 4.2.7 Consider, for example, a set of four procedures: $P_1 \cdots P_4$, with the following set of precedence relations: (P_1,P_3), (P_2,P_3), (P_2,P_4). The corresponding plan-graph is depicted in Figure 4.5(a). This plan-graph is consistent with five linearized plans: P_1-P_2-P_3-P_4, P_1-P_2-P_4-P_3, P_2-P_1-P_3-P_4, P_2-P_1-P_4-P_3, and P_2-P_4-P_1-P_3. However, if we then add another precedence constraint: (P_4,P_1) (Figure 4.5(b), then there is only one linearized plan consistent with the plan-graph: P_2-P_4-P_1-P_3. Finally, if we add a fifth constraint: (P_1,P_2) (Figure 4.5(c)), then a cycle is created (P_1, P_2, P_3) and there is no valid plan consistent with the plan-graph.

Given this characterization, the plan sketching algorithm presented next will reject plan-graphs with cycles. To restrict the number of plans considered, *preferences* can be specified as another type of precedence relation. Unlike constraints, satisfaction of a preference relation is optional; a plan will not be outright rejected if it does

not satisfy a given preference. I therefore also refer to preferences as "soft" constraints. However, such plan could eventually be rejected if there is a plan that clearly dominates it, i.e. that satisfies all of the preferences satisfied by the former, and more.

Definition 4.2.8 *A Plan-Graph (with Constraints and Preferences)*

Given a set of procedures PROCS, and sets of constraints C and preferences P on PROCS, a plan-graph is a directed graph G(PROCS,$C \cup P'$), where $P' \subseteq P$.

As before, such a plan-graph will be regarded valid only if it is acyclic. As just mentioned, multiple valid plan-graphs may exist, each corresponding to the inclusion of a different subset of preferences. These different plan-graphs may in fact be indistinguishable with respect to the partial order of procedures, e.g. when the ordering imposed by one arc can be inferred via transitivity from others, or of little or no importance, e.g. when the order in which two actions are taken is unimportant. A dominance relation can be defined over plan-graphs in which those with maximally (with respect to set-inclusion, cardinality, etc.) satisfied preferences are preferred. In TraumAID 2.0, preferences are notably used to linearize plans for the purpose of presentation.

Another type of constraint (or similarly preference): a *compatibility constraint*, can be implemented as an anti-symmetric pair of precedence constraints between the respective procedures. For example, if procedures P_1 and P_2 are incompatible, a pair of precedence relations (P_1,P_2), and (P_2,P_1) can be added to the set of constraints. Thus, whenever the two procedures are included in one plan, a cycle will emerge in its plan-graph forcing the rejection of such plan.

4.2.4 A Selection-and-Ordering Planning Algorithm

While it is conceptually useful to distinguish a selection and ordering *functions*, it was already established (cf. Section 1.2) that the two are mutually dependent on

one another, and thus must be *operationally* interleaved. The planning algorithm presented next interleaves selection and ordering.

Before actually describing the algorithm, a few more definitions are necessary. First, I will extend the Procedure Selection (PS) problem, defined in Section 4.2.2, with preferences on goals, and on use of procedures to address individual goals.

Definition 4.2.9 *Goal-Procedure Mapping (extended)*

Goal-procedure mappings used to enumerate alternative procedures which can be used to address individual goals (Definition 4.2.1). We can extend this definition with a ranking of alternative procedures (not necessarily strictly) by their respective preference *if the given goal was present alone.*

Definition 4.2.10 *The Extended Procedure Selection Problem*

An instance of a *Procedure Selection* (PS) problem includes a set of *goals*; a set of *procedures*; and a set of *goal-procedures* mappings as above. In addition, a *cost* function is individually specified for every procedure. Then, given a subset of goals G$\subseteq$GOALS, ranked by their *priority*, the problem is to find a subset of procedures P$\subseteq$PROCS that satisfies the following conditions:

1. *biased completeness criterion*: ideally, all goals should be addressed. A goal g will only *not* be addressed by P if all of its procedures are in conflict with all possible combinations of procedures that can be chosen to address goals ranked higher than or equal to g;

2. *individual preference optimality*: except for multiple-goal optimization and conflicts with goals of higher priority, a goal is always addressed by its procedure of preference; let g be a goal, and p be the procedure which addresses it in P, then if p addresses only g then if p' is an alternative procedure for g then either p is preferable to p' for g, or p' is incompatible with procedures chosen for goals ranked higher than g;

3. *biased cost optimality*: P has minimal cost among plans satisfying 1, and 2.

The purpose in defining the extended PS problem is to use the local preferences expressed by the respective ranking of goals, of the alternative procedures for a given goal, and of the individual procedures independently from their use, for the purpose of ranking complete plans by their desirability. Thus, the biased completeness criterion indicates that we prefer plans which are complete but, if bound to have an incomplete plan, will prefer a plan that is incomplete with respect to a less important goal. The individual preference optimality criterion indicates the desire to address each goal with the best procedure for that goal, but the willingness to sacrifice this desire if such procedure does not sit well with a preferred way of addressing a more important goal. Finally, the biased cost optimality criterion calls for reduction of overall cost via the use of procedures which address multiple goals at once.

The extended PS problem is clearly NP-Hard; it becomes identical to the original PS problem by simply choosing uniform preference and imposing no constraints. It introduces preferences on goals, and on the use of procedures for individual goals, to underconstrained plan-graphs. These preferences, or soft constraints, will also play a role in TraumAID 2.0's greedy algorithm.

The selection-and-ordering algorithm presented next also goes deeper in the level of specification of the desired activity. That is, it does not schedule procedures, but rather the actions that comprise them. In its representation, each procedure is broken into a sequence (or more generally partially ordered set) of actions.

Definition 4.2.11 *Procedure-Action Mapping*

Let PROCS $\stackrel{\text{def}}{=} \{\text{P}_i\}_{i=1}^{m}$ be a set of procedures, ACTIONS $\stackrel{\text{def}}{=} \{\text{A}_j\}_{j=1}^{l}$ a set of actions. A *procedure-action mapping* is a function from PROCS to the set of partially ordered subsets of ACTIONS, which indicates the action components of each procedure and their respective ordering requirements. The following relations can be defined:

$$\text{part-of}(a) = \{p \mid a \text{ is an action component of } p\}$$

$$\text{actions-of}(p) = \{a \mid p \in \text{part-of } (a)\ \}$$

$$\text{precedes-in}(p) = \{(a_1, a_2) \mid a_1 \text{ precedes } a_2 \text{ in } p\}$$

When a procedure is scheduled, its actions must keep their respective order, as indicated by the precedes-in relation. However, actions that belong to other procedures can be interleaved between them. Scheduling actions, rather than procedures, also gives rise to new opportunities in the *choice* of procedures. It often happens, for example, that procedures share component actions. In such cases, it is sometimes possible to reduce the overall cost of a plan by *merging* the procedures' actions, rather than simply carrying out one procedure after the other. In the PS problem, the *cost optimality* criterion must therefore be restated for *actions*.

Scheduling constraints and preferences can thus be determined at

1. the *goal* level: any action aimed at one goal must precede any action aimed at another;

2. the *procedure* level: all actions in one procedure must precede all actions in another; or

3. the *action* level: a particular action should precede another.

These levels of constraints give rise to certain difficulties with shared actions: it may not always be easy to determine the true role of such actions and the characteristics of which procedures they should inherit. In particular, it may be impossible to keep both the intra-procedural and inter-procedural constraints. (An example from the multiple trauma management domain is presented in Section 4.3.1.) This problem is partially solved in TraumAID 2.0 by allowing actions to carry their own characteristics that distinguish them from the procedures they participate in, and by allowing

additional scheduling information to be associated with each action (for example that is is not essential for it to meet a one type of *procedural* constraint or another).

The planning algorithm, described next, interleaves selection and ordering. Given the intractability of both sub-tasks, it is greedy in nature. It employs local backtracking only (with respect to the choice of procedure for the current goal), and uses goal-preference and goal-procedure preference as a heuristic.

Algorithm 4.2.12 *A Greedy Selection-and-Ordering Planning Algorithm*

1. Given a set of goals, they are first sorted by their priority;
2. Then, a plan graph Π is built incrementally by iteratively choosing γ, the highest-ranking goal that is not yet addressed by Π, and π the procedure of highest preference for γ. π's actions are then incorporated into Π with all the appropriate constraints. If that works, the next goal is being considered. If π cannot be incorporated into Π, then the algorithm backtracks to the next-best procedure for γ. If none of γ's procedures can be incorporated into Π then it is left unaddressed;
3. Finally, soft constraints (preferences) are added to Π.

This algorithm has been instantiated in TraumAID 2.0's planner, which details are presented next.

4.3 TraumAID 2.0's Planner

TraumAID 2.0's planner follows the PHP approach. The actual course of action it recommends is constructed incrementally, allowing for feedback and adaptation, and is made of the initial segments of multiple intermediate plans. In each cycle, an intermediate plan is constructed in a two stage process that is repeated if necessary:

1. A plan is first sketched using a selection-and-ordering algorithm;

2. This plan is then optimized to a horizon (currently consisting of only one action).

The specifics of each step is discussed next.

4.3.1 Plan Sketching

The plan sketching algorithm accepts as input a set of goals. Each of these goals are characterized by:

1. *Urgency.* Different goals need be addressed within different time frames. For example, instability and its sources need be addressed immediately whereas a definitive operation may sometimes be delayed until further examination is completed. In general, urgency denotes the time frame within which a goal has to be addressed. A 3-level urgency scale is currently used, corresponding to goals that have to be addressed within 2, 20, or 200 minutes.

2. *Priority.* In trauma management it is common to prioritize problems by their category. The ABC's of trauma management[7] call for the treatment of airway problems first, then those concerned with breathing, then circulation etc.

As per Section 4.2, the relationship between goals and procedures is given by *goal-procedure* mappings. Figure 4.6 illustrates one such mapping graphically: abdominal bleeding can be diagnosed using either a lavage or a CT scan. The preference, represented by the numbering on the arcs, is to use a lavage. That preference can be altered by the combination of goals presented, particularly if a CT scan is anyway required.

[7] ABC stands for Airway, Breathing, Circulation.

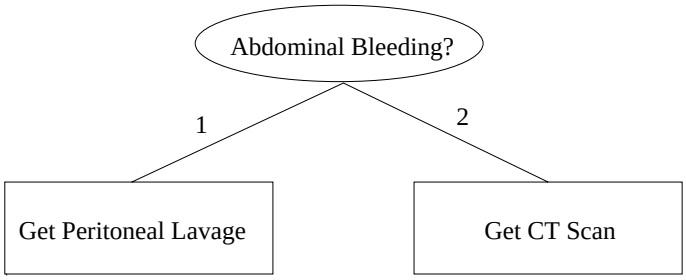


Figure 4.6: Alternative Procedures for Diagnosing Abdominal Bleeding

In TraumAID 2.0, each procedure is a *sequence* of actions, and is represented in a *procedure-action* mapping. Figure 4.7 illustrates one such mapping graphically: a standard treatment for simple pneumothorax includes, in that order, inserting a chest tube, observing the flow from it, and obtaining an X-ray to verify its placement and effectiveness.

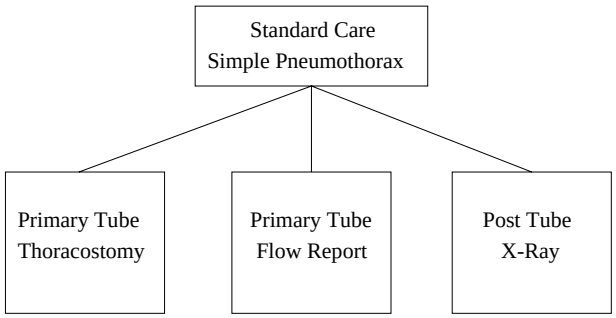


Figure 4.7: Actions for Standard Care Simple Pneumothorax

For each action, the following is represented:

1. *Duration.* In general, for scheduling purposes, it might be useful to represent the actual duration of actions. In TraumAID 2.0, a 3-level scale is used (categorizing actions that can be completed in 2, 20, or 200 minutes), corresponding to levels of urgency for goals. In the scheduling algorithm, these are used to constrain the use of lengthy procedures prior to the satisfaction of urgent goals.

2. *Cost/risk factor.* When there are multiple valid plans, one wishes to select the most preferable one. In general, preference is a function of the combination of

goals, actions, and the state of the patient. In TraumAID 2.0, a simple cost function is used to model preference. The cost of a plan is taken to be the total cost of its individual actions.

3. *Resources and logistic considerations.* In addition to time and physician attention, different actions may require different resources which may affect their use and respective scheduling. In TraumAID 2.0, certain actions are mandatorily performed only in certain sites, e.g. CT scan in the X-ray room, operations in the operating room, etc.

Following Algorithm 4.2.12, TraumAID 2.0's selection-and-ordering algorithm proceeds as follows:

Algorithm 4.3.1 *Interleaved Selection and Ordering in TraumAID 2.0*

1. The set of goals is first sorted by their *urgency* and *priority.*

2. Then, a plan graph Π is built incrementally by iteratively choosing γ, the highest-ranking goal that is not yet addressed by Π, and π the procedure of highest preference for γ.

   In incorporating π's actions into Π, the following constraints are imposed:

   (a) a lengthy action is not allowed before an action that is aimed at an urgent goal;

   (b) when placing an action in a certain site, it must follow all actions in preceding sites and precede all actions in later sites;

   (c) within a site, actions are ordered by the priority of their respective goals;

   (d) finally, actions are required to satisfy intra-procedural order.

   If a cycle is created by the constraint arcs, then the new procedure is incompatible with the previous plan, and is therefore rejected. The algorithm backtracks

to the next-best procedure for γ, or the next possible site for the current procedure, and try again. If none of the procedures for the given goal is compatible with the current plan, then this goal is left unaddressed.

Note that since goals were originally sorted by their urgency and priority, a goal is left unaddressed when its satisfaction is in an *apparent* conflict with the satisfaction of a more urgent-more important goal. Goal preference is thus a heuristic used by the greedy selection-and-ordering algorithm. Note however, that it is possible that a different choice for the latter *would have* allowed the satisfaction of the former, and so the greedy algorithm does *not* guarantee that the biased completeness criterion be satisfied whenever possible.

3. Finally, Π is linearized by incorporating soft constraints (preferences) in the form of a static, pre-defined, order on the set of actions.

Note that while the current use of resources is not very rich, there should be no methodological problem with extending it. For example, TraumAID 2.0's representation can be extended with resources that are either unavailable or are unlimited. Examples of such resources would be medical expertise or equipment. To adapt TraumAID for operation aboard a submarine, for example, all that had to be done was to mask out the procedures which require unavailable resources [99]. Other scheduling techniques would allow planning with limited resources.

As already discussed in Section 4.2, reasoning in the action level allows procedures to be *merged*, with the ordering of the urgent sub-parts of multiple procedures before their less-urgent parts. It also allows unifying common actions. Consider, for instance the standard procedure for treating a hemothorax condition. It comprises a sequence of five actions: administration of antibiotics, setup of an auto-tranfusion device, thoracostomy and leakage (flow) report, and post-tube X-ray. Figure 4.8 depicts the plan graph (without soft constraints) for a patient with a hemothorax on both sides.

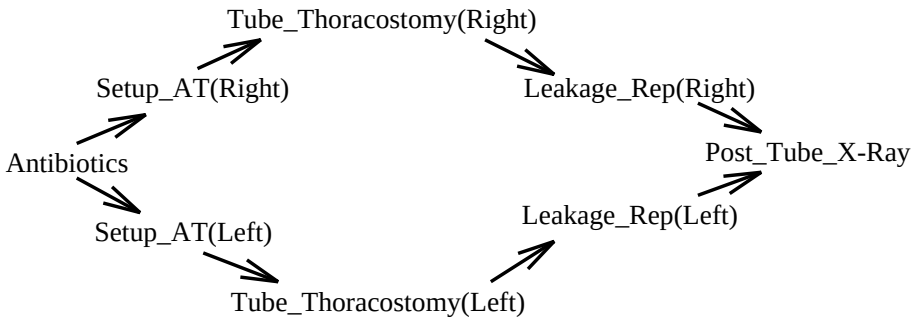


Figure 4.8: Plan Graph for Patient with Hemothoraces on Both Sides

Reasoning in the action level also presents some challenges. For example, when an action is shared by procedures which address goals of distinct priority, one has to decide which of the features of these goals this action inherits. In the case of a patient with both a pneumothorax (Figure 4.7) and a hemothorax, for instance, a conflict may arise with the respective scheduling of the antibiotics and the thoracostomy. On one hand, the antibiotics precedes the thoracostomy within the hemothorax procedure. On the other hand, the thoracostomy should precede the antibiotics as it addresses a problem of higher priority (the pneumothorax). To partially circumvent this problem, actions are each allowed to carry its own characteristics and scheduling information. For example, we do not require that the administration of antibiotics satisfies priority constraints.

4.3.2 Plan Optimization

The plan sketching algorithm just described is a greedy one, and thus cannot guarantee an optimal plan. To improve the quality of its plans, in addition to the ECM-based adaptation, TraumAID 2.0's planner performs a one-action deep optimization. In that optimization step, potential replacements of the procedure to which the first action in the plan belongs, with procedures which address other goals as well, are considered. The effects of various replacements are evaluated using the overall *cost* as a metric.

I distinguish *general* optimizers from *domain-specific* ones[8]. I have identified two general scenarios in which, being greedy, TraumAID 2.0's selection-and-ordering algorithm makes wrong coverage decisions. Those are depicted in Figure 4.9 where goals are indicated by circles and procedures by rectangles. The numbers on the arcs represent order of preference. For simplicity, assume that all procedures are equal in terms of their risk, cost etc. In both situations, the greedy algorithm will select procedure **1** for addressing goal **A** and procedure **2** for goal **B**. However, a quick glance at the first example shows that procedures **1** and **2** can be both replaced with procedure **3**. Similarly, in the second example, procedure **2** can serve both purposes. To address these shortcomings, TraumAID 2.0's optimizer considers all possible replacements for the currently optimized procedure.

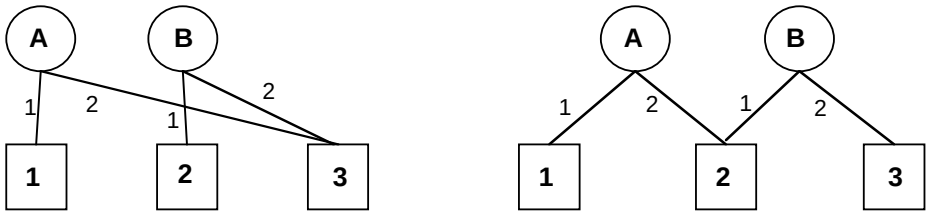


Figure 4.9: Selection Weaknesses in the Greedy Algorithm

So far, in addition to these general optimizers, two domain-specific optimizers were needed. The first optimizer calls to avoid inserting a prophylactic tube in patients scheduled to undergo an operation soon[9]. Like the general ones, this optimizer can also be represented as a subsumption operator.

The second optimizer, called *Rush-to-OR*, applies when a patient is in the emergency room (ER) and an urgent procedure must be done in the operating room (OR) within a restricted time frame. At such point, even if planned management has some actions that may be carried out in the ER, it is preferable to immediately transfer the patient

[8]Note that by general I mean domain-independent; those optimizers are still representation- and algorithm-dependent.

[9]Anecdotally, this optimization was *not* well-received by some surgeons (see Section 6.2).

to the OR, and then continue management there. (All actions schedulable in the ER can also be scheduled in the OR.) The *Rush-to-OR* optimizer takes a plan after all subsumption operators have been applied.

4.4 Summary

In this chapter, I have presented a planning framework for the ECM architecture which I called Progressive Horizon Planning (PHP). In the PHP framework, intermediate plans are constructed in a partial-global manner, followed partially, and then reconstructed to reflect new information and goals. Intermediate plans are constructed via a combination of plan sketching and partial optimization.

I have also presented a planning paradigm in which the planning task is functionally factored into *choice* of means for addressing a combination of goals, and *ordering* chosen means into a single overall plan. I have presented an algorithm which interleaves the two functions.

This PHP framework is implemented in TraumAID 2.0's planner, where an instantiation of the selection-and-ordering algorithm is used for plan sketching.

Chapter 5

Use of the ECM Architecture and Reasoning Components

5.1 Overview

Most of the discussion so far has focused on the function, but not the use, of the ECM architecture and its reasoning components. Next, I propose a framework for "programming" the ECM architecture's components to produce a desired behavior. In particular, in accordance with the suggested decomposition of reasoning in the ECM architecture, I propose

1. that *local* patterns of behavior (or strategies) be *explicitly* encoded
   (a) in GDD rules, using goals as milestones and choice points, and using goal-setting and evidential rules to represent progress within a local strategy;
   (b) in pre-defined local procedures: sequences of actions that can be used to locally satisfy a given goal; and
   (c) in mappings from goals to a set of alternative procedures, ordered by their respective local preference for the given goal (i.e. independently of other needs);

2. that constraints and preferences on *combinations* of goals, procedures, and actions, be specified as much as possible in the form of general principles; and finally

3. that given a combination of goals, these principles be used by the planner to *implicitly* merge a corresponding combination of local strategies, as necessary.

In this chapter, I put together the diagnostic reasoning and planning frameworks described in Chapters 3 and 4. I will first discuss local strategies that can be encoded explicitly in GDD rules and goal-procedure mappings, and then mechanisms for combining a number of such strategies on-line. I conclude by pointing to potential uses of this methodology for managing JR's case (cf. Section 1.2).

5.2 Local Strategies

In programming artificial agents, we try to provide them with a *strategy* – the capability to respond appropriately to each of the situations in which they may find themselves[1].

A strategy can be represented *explicitly*, e.g. as an enumeration of states and responses; *implicitly*, e.g. as a computation procedure that can be applied to a state's representation to compute a response; or as a *combination* of explicit and implicit representations. In general, different representations of strategies follow from different tradeoffs between the time and space needed to construct, represent, and use a given strategy. Thus, in choosing a representation, one has to consider the tradeoffs that are appropriate for the problem at hand.

Consider for example the AB-space of Chapter 3, where states are attitude-belief assignments and where transitions correspond to changes in attitude or in belief

[1] I take strategy as commonly defined in the game theory literature, e.g. "a strategy is a rule that tells [the player] which actions to choose at each instant of the game given his information set ... a player's strategy is a complete set of instructions for him, which tells him what actions to pick in every conceivable situation, even if he does not expect to reach that situation." [128].

that are either volunteered, inferred, or acquired via action. Taking attitude-beliefs to be the basic states on which a strategy has to be defined, there is more than one way in which this can be done. On one extreme, one can explicitly enumerate all attitude-belief states that the agent could ever be at, designating a response (an inference or an action) for each. On the other extreme, given an attitude-belief, a response can each time and again be computed on-line from first principles.

In many domains, trauma management included, an agent's task can be broken down into a combination of several sub-tasks, each of which is *not* very complex. The complexity of the overall strategy in such domains is thus the result of the potential *interactions* between sub-strategies. I propose that in such domains it may be useful to explicitly specify each of the sub-strategies, while using more implicit reasoning to compute response to combinations thereof. This approach is particularly useful in domains in which a large number of sub-strategies are potentially applicable but only a few, albeit possibly different ones each time, are likely to be concurrently pursued.

I call each of these sub-strategies a *local* strategy. In the trauma management domain, strategies can be localized around smaller problems. For example, Section 3.7 enumerates states[2] which are of relevance to the diagnosis and treatment of a tension pneumothorax. In other domains, local strategies can be organized around a certain resource or expertise, e.g. planning a series of X-rays vs. planning an operation, or describing work done by a physician vs. that done by a nurse. Localization is often domain-specific, and one may therefore have to rely on an expert's intimate knowledge of the domain. More generally, the use of localized strategies in the proposed methodology is a simple matter of divide-and-conquer and so it can also accommodate automatically generated sub-plans (as in [162, 94, 48]), as well as combinations of automatically generated sub-plans and explicitly specified sub-strategies.

An important distinction of our local strategies is that while inference-responses are

[2]To be precise, rules enumerate equivalence classes on states. They do not describe a single state, but rather the set of states which meet the rule's antecedents.

explicitly specified in GDD rules, action-responses are *not*. Instead, local strategies specify *goal*-responses, and a set of alternative procedures/actions that can be used with the *intention* of satisfying these goals (and thereby bringing about the local strategy's next state, or conclusion). There are two issues here. First, it is not required that action-responses, eventually computed from goal-responses, necessarily achieve a pre-determined effect. While it is *intended* that a procedure satisfy the goal(s) for which it was chosen, it is also acknowledged that an action can result in one of a number of outcomes (therefore the requirement that a local strategy be *closed* with respect to *all* such outcomes). Second, using goals one can specify *alternative* choices of action for each, and state one's preference with regard to their choice. These *choice points* in a strategy add to the planner's ability to merge several local strategies, particularly to its capability of taking advantage of potential synergies. In addition, as will soon be demonstrated, the use of goals may sometimes pay off relieving some of the computational complexity of planning. Next, I discuss merging local strategies, in the goal- and procedure/action-level.

5.3 Implicitly Combining Local Strategies

The way in which local strategies are combined in the ECM architecture is implicit in two ways. First, the combined strategy is never represented explicitly. Rather, it is constructed on the fly, as it is executed. Second, for the most part, what is encoded are combination *principles*, as opposed to a recipe as to how to merge any particular combination of strategies.

Strategies can be combined in two levels:

1. In the *goal-level*, one can specify when a certain goal suppresses, inhibits, or is prioritized with respect to another, and when a certain *combination* of goals can be replaced with a *generalizing* goal or set of goals;

2. In the *procedure/action-level*, one can specify a preference for one procedure over another with respect to a certain goal; a preference for a certain combination of procedures for a certain combination of goals; and compatibility and ordering constraints between two or more procedures or actions.

5.3.1 Combining in the Goal-Level

Many times, local strategies can be combined at the goal level, without having to reason about possible choices of action for these goals. Of particular interest are *negative* relationships between goals that would result in a reduction in the number of goals that need subsequently be considered by the planner. Consider for example, the following relationships:

- Suppression: The emergence of one goal subsumes another. This can be because one goal is a specialization of another, e.g. need for an abdominal operation for unknown reason versus need for an abdominal operation to repair the duodenum; it can be because a certain *treatment* subsumes another, or render it inappropriate, e.g. an emergency room thoracotomy will subsume any other diagnosis and/or treatment; with diagnostic goals, it can be because *any* information that can be acquired by *any* action aimed at one goal would also satisfy the other goal, e.g. an exploratory laparotomy and some of the abdominal investigations.

- Inhibition or prioritization: The *current* pursuit of one goal inhibits or delays the pursuit of another, e.g. pursuit of abdominal bleeding as a possible cause of shock is inhibited in some patients if a tension pneumothorax possibility has not yet been pursued. Note that the only difference between inhibition and suppression is that an inhibited goal may be resumed after the inhibiting goal is satisfied. A special case is prioritization, when one goal shall be pursued before

another, e.g. a pneumothorax should often be pursued prior to an esophageal injury.

Such relationships are encoded in the ECM architecture using *goal inhibition* clauses which are then expanded into GDD rules, e.g.

- the desire to suppress g_2 in the presence of g_1 is expressed as:
$$inhibit(g_2) = relevant(g_1)$$
- the desire that g_1 inhibits g_2, until or unless g_1 is determined negative is expressed as:
$$inhibit(g_2) = relevant(g_1) \wedge \neg false(g_1)$$
- the desire to prioritize the pursuit of g_1 over that of g_2 is expressed as:
$$inhibit(g_2) = relevant(g_1) \wedge \neg known(g_1)$$

Basic prioritization information is encoded in goals' features, which are used to determine their respective ordering by the planner. For example, since a pneumothorax's *priority* is *airway*, its diagnosis and treatment will usually precede that of an esophageal injury which priority rank is *contamination*. In TraumAID 2.0's planning algorithm (Algorithm 4.3.1), such priorities translate into (a) preferred choice, in that the greedy algorithm first chooses a procedure for higher-ranking goals, and (b) precedence, in that actions aimed at goals of higher priority are often scheduled prior to actions aimed at goals of a lower one.

5.3.2 Combining in the Procedure/Action-Level

In the ECM planner, procedures and actions for concurrently pursued local strategies are merged in the selection-and-ordering algorithm using the following combination principles:

1. Preference for addressing one goal versus another is encoded in the principles by which goals are first sorted. In TraumAID 2.0: a goal's urgency and priority.

2. Preference among alternative procedures for a given goal is specified in a goal-procedure mapping for each goal individually.

3. Some restricted forms of preference among *combinations* of procedures for *combinations* of goals are specified via the cost measure used by the parsimonious set-covering model. More complicated forms may require explicit specification either in GDD rules which manipulate goals, or in optimization transformations which are later applied to the plan sketch.

4. Constraints and preferences on co-presence and ordering of several procedures within a plan can be specified for the scheduling algorithm. TraumAID 2.0 uses *constraints* based on the respective urgency and priority of the corresponding goals and on the site in which a procedure is selected to be performed at; a general linearizing *preference* is also used.

5.4 Example

Consider again JR's management (Section 1.2), presenting an interplay of several local strategies such as diagnosis and treatment of a tension pneumothorax (cf. Section 3.7), diagnosis and treatment of a pericardial tamponade (cf. Example 3.3.6), general abdominal examination, treatment of a hemothorax, and others.

First notice that the different strategies are triggered at different times during management and that each proceeds in a different pace. Also note that the combined strategy is *not* represented per se, but is rather implicitly and incrementally constructed via an on-going mediation between the local strategies.

Most of the explicit and implicit encoding mechanisms are represented in this single case, in particular:

1. The goal of inserting a chest tube for the right hemothorax is *suppressed* by the goal of operating the chest;

2. The goals of investigating the pericardial sac and abdomen for causes of shock are *inhibited/prioritized* until after the investigation of a tension pneumothorax as an alternative explanation is completed;

3. Their respective features indicate that the goal of diagnosing a tension pneumothorax is more urgent and has a higher priority (airway vs. circulation) than the goal of investigating the abdomen;

4. The goal of diagnosing a pericardial tamponade can either be addressed with an ultrasound, or via a needle aspiration. This order of *preference* for alternative procedures is local though, and is often violated in favor of compatibility with other needs. In fact, in JR's case a needle is used due to the incompatibility of the ultrasound procedure with the present urgency;

5. Preference for a *parsimonious combination* of procedures for the combination of goals is exemplified by the choice of a bilateral operation as a single operation that can be used to gain access to both the heart and the left chest;

6. Finally, the use of *constraints* in the on-line merging of local strategies is exemplified by the following:

    - Constraints based on *goal features* are exemplified by ordering the aspiration of the chest prior to the aspiration of the pericardial sac, and also, as per (3), by ordering it prior to the abdominal X-ray studies;
    - Constraints based on *logistic considerations* are exemplified by scheduling the abdominal X-ray prior to the thoracic arteriogram. The arteriogram is later discarded, but at first it is scheduled later because it requires access to machinery that is only available in the X-ray room;

- Constraints based on *time/urgency* issues are exemplified by the selection of an *otherwise less-preferable* needle aspiration as a means of diagnosing a pericardial tamponade. The ultrasound, which is usually preferable, conflicts with the urgency of this and other goals;
- Constraints that result in an *total incompatibility* are exemplified in that the arteriogram, as well as the investigation of a potential esophageal injury, are not pursued at all once JR is transferred to the operating room.

Chapter 6

TraumAID 2.0 – Implementation and Empirical Results

TraumAID 2.0 is a consultation system for the multiple trauma mangement domain. It is a successor to TraumAID 1.0 (cf. Section 1.2.3), and was implemented in Common Lisp on a Symbolics Lisp machine. It was recently ported by Jonathan Kaye to a Sun-based X-windows environment as well as to a Macintosh platform.

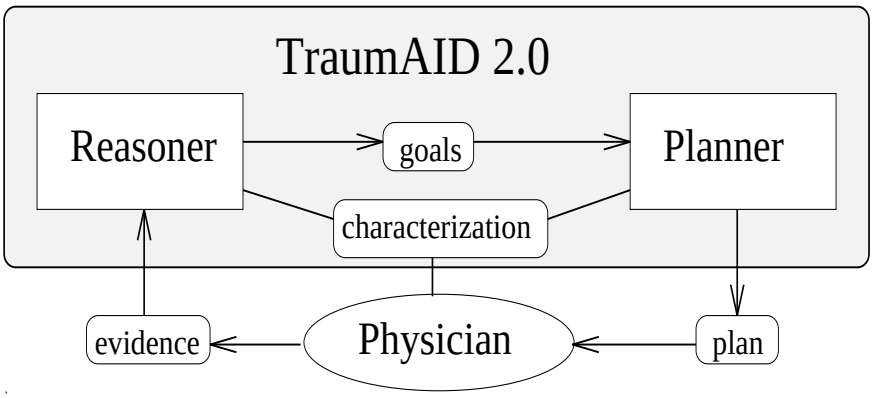


Figure 6.1: ECM architecture: basic cycle of Reasoning, Planning and Action

TraumAID 2.0 implements the ECM architecture (Figure 6.1). Its diagnostic reasoning follows a Goal-Directed Diagnosis framework (see Section 3.9). It uses a Progressive Horizon Planner with a Selection-and-Ordering sketching algorithm (see

Section 4.3).

This chapter follows step by step as a management plan is being developed for JR's case (cf. Section 1.2) by TraumAID 2.0. Then, it presents results from an empirical study in which management plans produced by TraumAID 2.0 were compared to those produced by its predecessor TraumAID 1.0 and to the actual care delivered to these patients in a trauma center. Finally, it pauses on a few particular cases which illustrate differences between TraumAID 2.0 and TraumAID 1.0.

6.1 TraumAID 2.0 – Example Case

Section 1.2 presented the case of JR, a trauma patient who was admitted to the emergency room in an unstable condition as a result of two gun shot wounds to the chest. In this section, I present snapshots from TraumAID 2.0's interface as it handles JR's case.

TraumAID 2.0's X-Windows interface uses a number of windows to convey information and interact with the physician. Figure 6.2 presents most of them, as appearing at the time when JR's management is complete. Most of the computer-guided interaction occurs on windows that pop up with a question or instruction and disappear immediately after the input is recorded in the appropriate summary windows. There are two command panels, shown at the upper left side of the screen. A *Conclusions* window presents conclusions and suspicions. Two other windows present *goals*: *Projected* and *Pursued*. (Note that even though JR's management has already been concluded, four goals were left unattended, subsumed by the definitive surgical procedure.) Another window, labeled *Assumptions* presents evidence which was provided to the program throughout the management. Procedures that were *performed* are recorded in another window. In addition to these windows, and the computer-guided interaction window, a plan window will also appear during management. Since the initial definitive management is complete, the plan is empty.

Figure 6.2: JR's Management Concluded

Let us begin at the point in which JR presented to the emergency room. After the two wounds are reported to TraumAID 2.0, it begins asking *all* relevant *bed side* questions. Bed side questions are pieces of information which are not associated with a costly, risky, painful, or time consuming procedures, and which are therefore not *planned* for but rather are surfaced as they are determined to be of possible relevance. TraumAID 2.0 asks those in a pre-determined order, grouping related questions for coherence. JR's findings include: shock, distended neck veins, decreased breath sounds on both sides, and muffled heart sounds. JR is not unconscious or obtunded, does not suffer from hemoptysis, and has no ileus or abdominal scarring. His pulses are all normal. Figure 6.3 presents the *Assumptions* window after the initial bed-side questions.

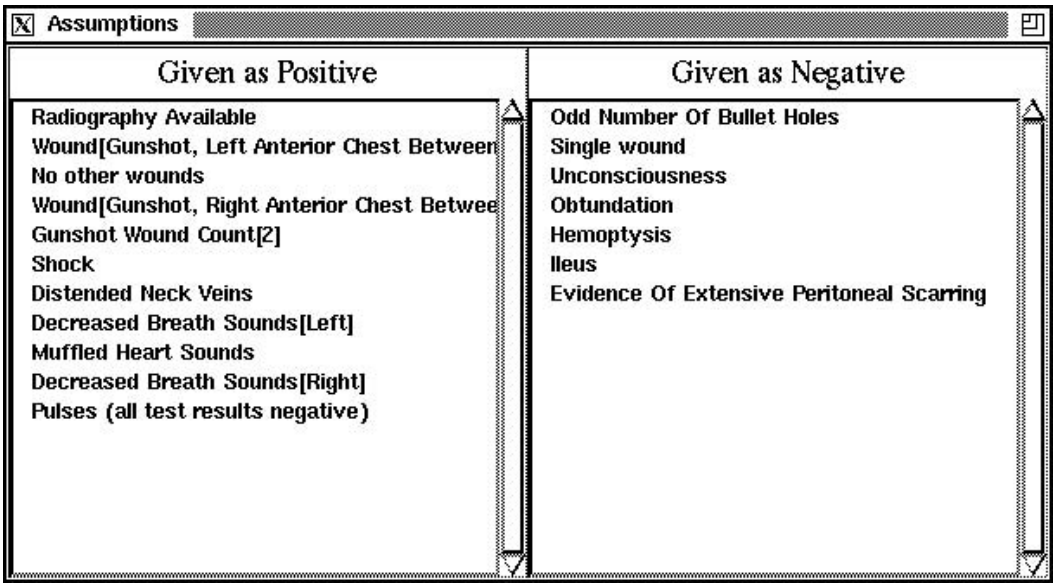


Figure 6.3: After Initial Bed-Side Questions

While no diagnoses other than a penetration of the chest are made, a number of goals, mostly diagnostic, are instantiated. Figure 6.4 presents the *Projected Goals* window.

TraumAID 2.0's *Plan* is presented in Figure 6.5. The plan represents a partial order on the selected set of actions, which is then linearized when presented according to a

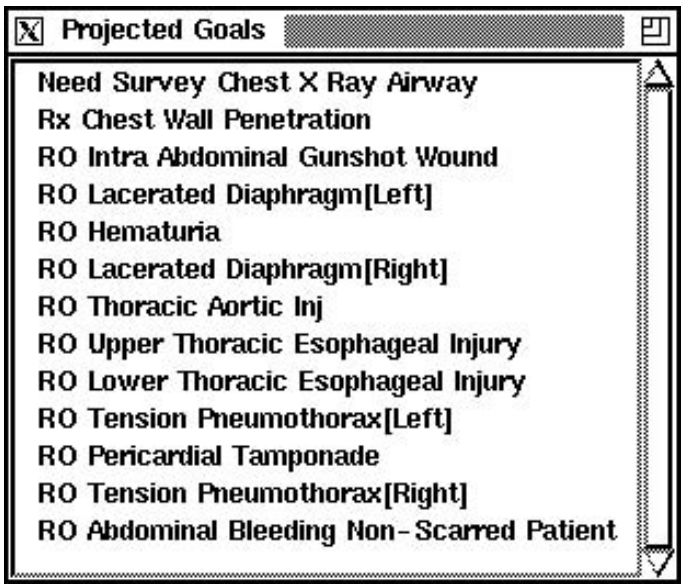


Figure 6.4: Initial Projected Goals

pre-determined fixed order. It is organized by the *hospital sites* in which actions are recommended. Actions are further ordered by their respective urgency and priority and by their order in their respective procedures. Urgent actions are marked with an asterisk.

The plan first calls for covering the wound. Although this is not an urgent action, it can be done quickly enough to allow its scheduling prior to urgent ones. This ordering is done in the linearization step, and is based on the *soft* ordering constraints. In the partial order based on the hard constraints, wound covering is *unordered* with respect to the urgent actions.

The first important goal, per the plan, is locating the cause of shock. Three actions are directly aimed at this goal, the aspiration of both sides of the chest; and the aspiration of the pericardial sac, which is ordered after the former because it is related to circulation problems and not airway. Notably, another action which can also be used for that purpose appears on the plan, although for other purposes: the survey chest X-ray, a standard procedure in chest injuries, can be used to diagnose a tension pneumothorax, however is inappropriate in unstable patients. The abdominal X-ray

is used to diagnose abdominal injury. The preferred diagnostic procedure for *shock* from abdominal bleeding, a peritoneal lavage, appears late in the plan because the urgent *goal* of diagnosing shock of abdominal origin is inhibited by the investigation of a tension pneumothorax and/or a pericardial tamponade. A few more studies: a urinalysis, an arteriogram, and a barium swallowing study, are planned for later stages.

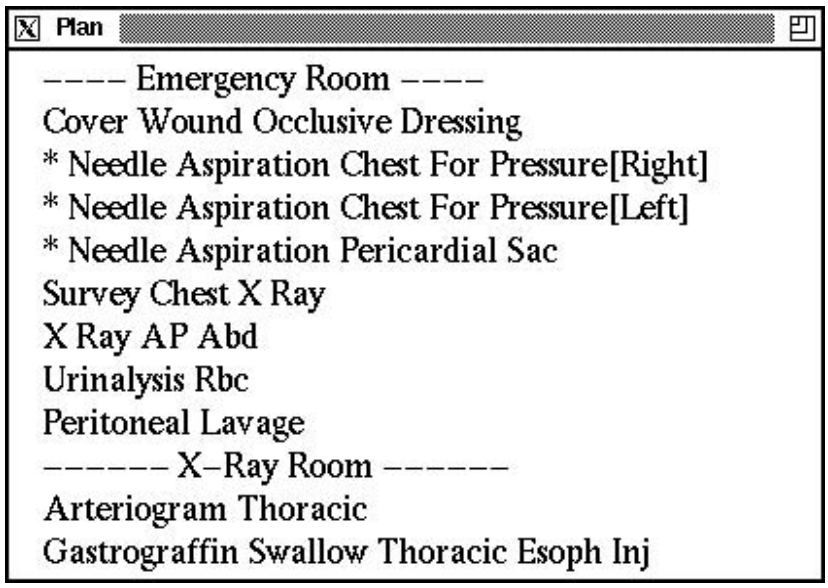


Figure 6.5: Plan after Initial Assessment

The needle aspiration of the chest revealed a tension pneumothorax on the left side, but JR remained in shock even after the aspiration has relieved the pressure in the chest cavity. TraumAID 2.0 recommended aspirating the pericardial sac, immediately after inserting a chest tube to prevent a deterioration in JR's condition. The aspiration of the pericardial sac revealed a pericardial tamponade. Figure 6.6 presents the current plan.

Note that, at this point, given that a need for an urgent heart operation was identified, JR was moved to the operating room for the remainder of the management. Also note that all lengthy studies, e.g. barium swallow and arteriogram, were removed from the plan. Note the interleaving of the pneumothorax treatment strategy,

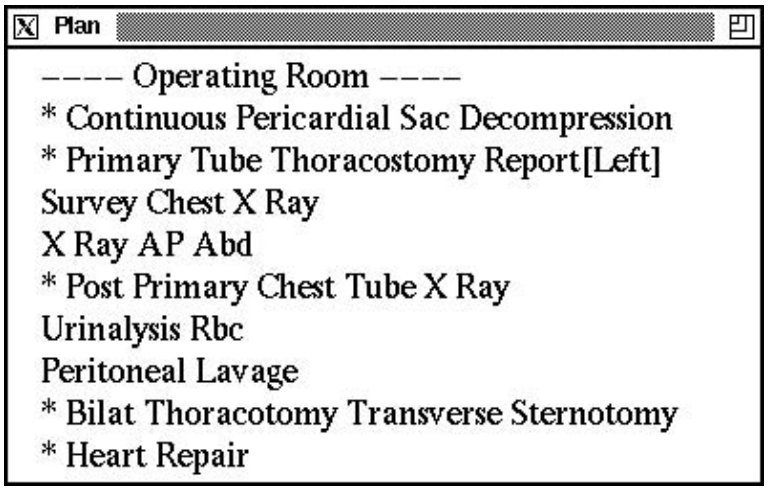


Figure 6.6: The Plan after Diagnosis of Pericardial Tamponade

which includes the chest tube insertion and subsequent flow monitoring and post-tube X-ray, and the treatment for the pericardial tamponade condition, including the decompression and the heart repair itself.

JR's pericardial sac was decompressed. Then, the flow off the chest tube has revealed a massive hemothorax. The next plan, presented in Figure 6.7 prepares for a left thoracotomy.

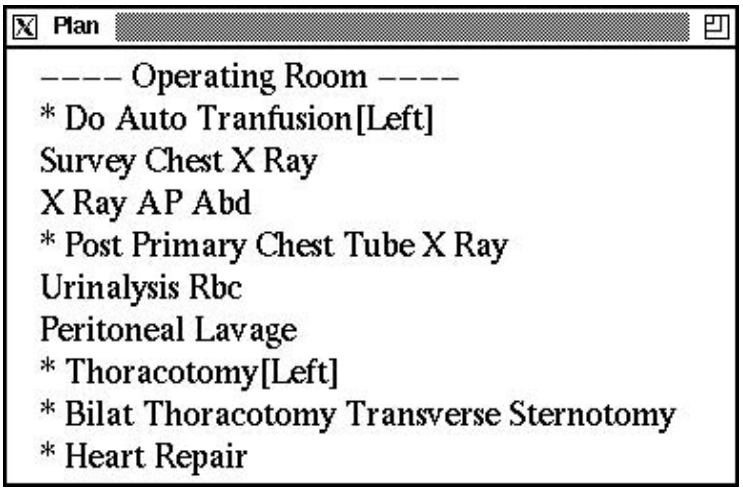


Figure 6.7: The Plan after Diagnosis of a Massive Hemothorax

Figure 6.8: A Survey Chest X-Ray

Note that, at the moment, both a left thoracotomy *and* a bilateral thoracotomy are planned. TraumAID 2.0's planner knows that the left thoracotomy is subsumed by the bilateral operation, but will only consider that when they are included within its horizon. In the meanwhile, the autotranfusion device is installed at the chest tube, and a survey chest X-ray is taken (Figure 6.8).

The survey chest X-ray reveals an additional hemothorax on the *right* side. It shows *one* bullet only, triggering the suspicion that the other bullet is in the abdomen. Figure 6.9 presents the new plan.

The new plan aims at locating that other bullet in the abdomen through an abdominal X-ray, or maybe finding another exit hole. It also schedules a lateral chest X-ray

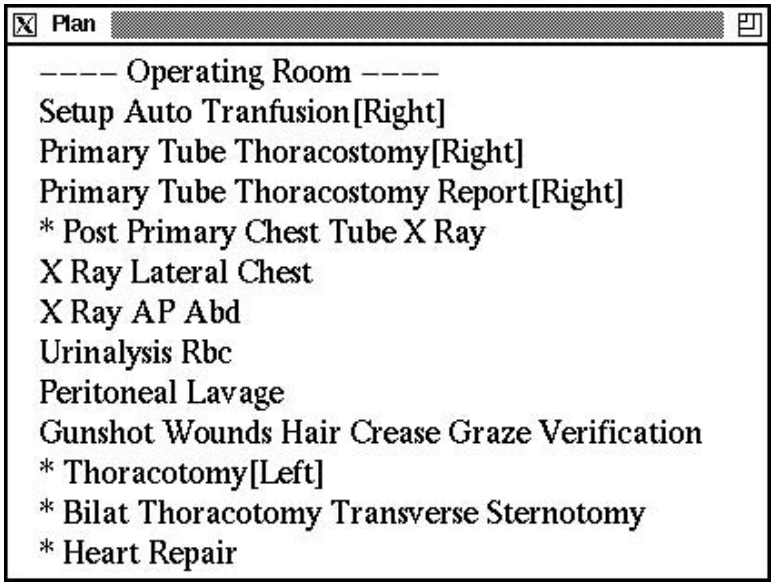


Figure 6.9: Plan After Chest X-Ray

to locate the exact location of the bullet in the chest, and calls for insertion of a chest tube in JR's right chest to treat the hemothorax condition. This chest tube, however, will not be inserted; when it gets within the optimization horizon, one of the planner's optimizers will discard it due to the proximity of the chest operation. After the lateral chest X-ray reveals that the bullet is lodged at the vicinity of the spine, a few more bed-side questions are asked to rule out neurological damage.

The abdominal X-ray, taken next, shows the missing bullet in the abdominal cavity, and a need for a laparotomy is established. The next plan, depicted in Figure 6.10, is not looking for the bullet anymore. The previously scheduled peritoneal lavage has been removed from the plan given that a need for a laparotomy has already been established.

After a urinalysis is determined negative, the chest operations are merged into a single bilateral surgery (Figure 6.11). Finally, an exploratory laparotomy is performed, in which both sides of the diaphragm are checked for penetration and repaired if necessary.

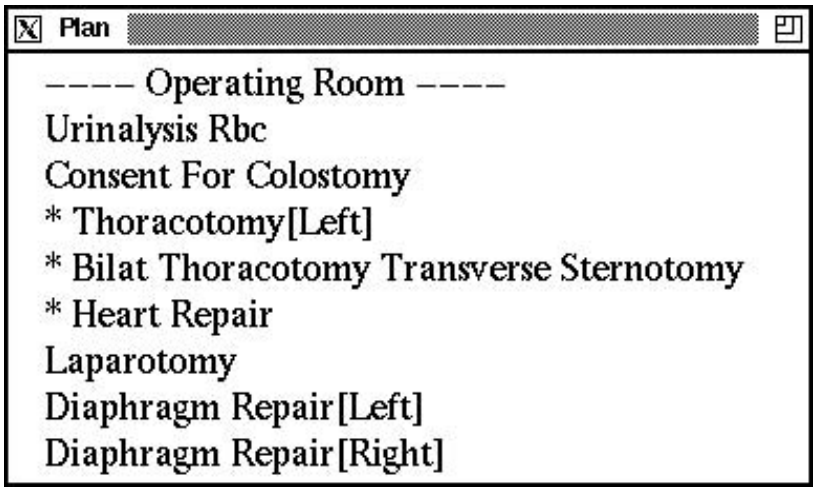


Figure 6.10: Plan After Need for Laparotomy is Established

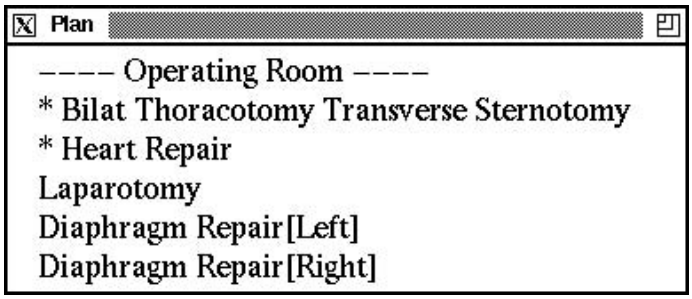


Figure 6.11: Plan Combining Chest Surgeries

6.2 A Comparison of Computer-Generated Protocols and Actual Care

Subsequent to the development of TraumAID 2.0, a retrospective study was carried out in which actual management plans were compared to those that would have been recommended by TraumAID 2.0 and TraumAID 1.0 [20]. In a blinded test, three trauma surgeons from the Medical College of Pennsylvania, who are *not* otherwise associated with the TraumAID project, were asked to compare actual care provided in 97 real trauma cases to alternative management plans generated by TraumAID 2.0 and by TraumAID 1.0.

Over a period of 15 months, 97 trauma cases of otherwise healthy adults who presented to the Trauma Center of the Medical College of Pennsylvania were collected. Each of the patients was stable or stabilized, and suffered gunshot or stab wounds confined to the chest and abdomen. Patients who were never successfully resuscitated to receive definitive care were excluded from the study. Also excluded were patients who suffered injuries other than gunshot or stab wounds and/or suffered serious wounds to other parts of the body.

Management plans were transcribed to have the same "look and feel". For control purposes, judges were asked to try to identify whether a given plan was generated by a computer. A set of 4 cases was selected by a consultant to be discussed with the judges prior to them being presented with the real cases. These cases were analyzed by a regional quality-assurance committee and comprise one case in which care is clearly acceptable, one case in which care is clearly unacceptable, one case which was acceptable by majority vote, and one which was unacceptable by a majority vote. Further, another set of 5 cases which the judges have analyzed in the past as part of a previous validation study, were presented to the judges in order to test consistency. To further check consistency, 10% of the cases were resampled and presented again to the judges. Finally, cases in which there was no consensus between the judges regarding acceptability were presented to them again for reconsideration.

Each judge was first asked to rate each management as being *Acceptable* (ACC), *Acceptable with no errors of major consequence* (NME), *Acceptable with reservations* (RES), or *Unacceptable* (UNACC).

Program	ACC	NME	RES	UNACC	GPA
Actual Care	3	39	41	14	2.39
TraumAID 1.0	3	39	50	5	2.44
TraumAID 2.0	17	53	23	4	2.85

Figure 6.12: Absolute Rating and GPA

Figure 6.12 summarizes results of the absolute rating of each of the 97 cases. The grade for each case management was averaged over the three judges, with a grade of 4 for ACC rating, 3 for NME, 2 for RES, and 1 for an UNACC rating. For example, a case graded by the three judges as ACC, NME, and UNACC respectively would have an overall grade of $(4 + 3 + 1)/3 = 2.67$. In the table, cases with overall grade of 3.5 or more are counted as ACC, between 2.5 and 3.5 as NME, between 1.5 and 2.5 as RES, and below 1.5 as UNACC. The figure denotes the number of cases falling into each category for each of the three management alternatives, and the overall Grade Point Average (GPA) for each of the management alternatives, i.e. the average over all 97 cases. Figure 6.13 presents these results graphically in a cumulative histogram.

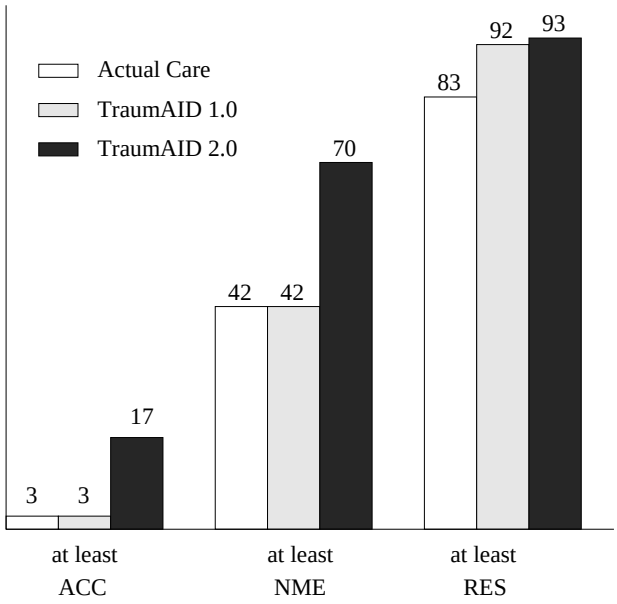


Figure 6.13: Cumulative Absolute Rating

Figure 6.14 presents a two-way comparison of the categorical ratings of management plans from TraumAID 2.0 and the actual care. In 51 cases, TraumAID 2.0 was ranked better than the actual care by at least one category-difference versus 13 cases in which the actual care was ranked better ($p < 10^{-6}$ by binomial test). In 14 cases, TraumAID 2.0 was ranked better by at least two categories versus 4 cases in

		TraumAID 2.0			
		ACC	NME	RES	UNACC	Total
Actual Care	ACC	0	1	2	0	3
	NME	8	22	7	2	39
	RES	5	25	10	1	41
	UNACC	4	5	4	1	14
	Total	17	53	23	4	97

Figure 6.14: Comparison of Absolute Rating of TraumAID 2.0 and Actual Care

which the actual care was ranked better by two categories (p <0.0155). Finally, in 4 cases, TraumAID 2.0 was ranked better by three categories (p =0.0625), i.e. fully acceptable versus fully unacceptable. The opposite did not occur in any of the cases.

Figure 6.15 presents a similar table for TraumAID 2.0 and TraumAID 1.0. In 35 cases, TraumAID 2.0 was ranked better than TraumAID 1.0 by at least one category-difference versus only 3 cases in which TraumAID 1.0 was ranked better (p <10^{-7}). TraumAID 2.0 was ranked better than TraumAID 1.0 by at least two categories in 10 cases, and was rated totally acceptable in one case in which TraumAID 1.0 was rated as totally unacceptable (p <0.001). In none of the cases, was TraumAID 1.0 ranked better than TraumAID 2.0 by more than one category-difference.

Judges were also asked to pairwise rank different management plans for each case by preference. These comparisons were required to be consistent with their own categorical ranking, i.e. if a certain plan was ranked NME and another was ranked RES, then the former plan must be preferable to the latter. Any preference could be

TraumAID 1.0 \ TraumAID 2.0	ACC	NME	RES	UNACC	Total
ACC	3	0	0	0	3
NME	4	33	2	0	39
RES	9	20	20	1	50
UNACC	1	0	1	3	5
Total	17	53	23	4	97

Figure 6.15: Comparison of Absolute Rating of TraumAID 2.0 and TraumAID 1.0

indicated within a category, i.e. if two plans were both ranked ACC, then the judge was asked to either rank them, or note them as equally preferable.

In paired comparisons of all plans of all 97 cases, the judges had a significant preference for TraumAID 2.0 plans over actual plans by a ratio of 64 to 17 with 16 ties ($p < 10^{-7}$). They had a significant preference for TraumAID 2.0 plans over TraumAID 1.0 plans by a ratio of 62 to 9 with 26 ties ($p < 10^{-10}$). Their preference for TraumAID 1.0 plans to actual plans by a ratio of 43 to 38 with 16 ties was not statistically significant ($p > 0.32$). Figure 6.16 summarizes these results.

Figure 6.17 presents a more refined comparison of the management plans of TraumAID 2.0 and the actual care. Cases are grouped by their categorical rating for each of the two plans. Within a category-product, cases are grouped by the preference margin, computed as the difference between the number of judges preferring one management plan and those preferring the other. Thus if one management plan was preferred by two judges, and the other preferred by the third judge, then the former

		Preferred Over		
		TraumAID 1.0	TraumAID 2.0	Actual Care
Preferred	TraumAID 1.0	■	9	43
	TraumAID 2.0	62	■	64
	Actual Care	38	17	■

Figure 6.16: Preference Ranking for All Three Management Plans

plan is preferred by a margin of 1. If the latter judge liked both management plans equally, then the margin would have been 2.

Note that on rare occasions, there is an inconsistency between the categorical ranking and the voted preference. In particular, notice the one case in which TraumAID 2.0 was ranked better than the actual care (NME vs. RES) but the overall preference favored the actual care. This phenomenon is a result of a disagreement among the judges in which two judges *slightly* favor one management plan, whereas the third judge *strongly* prefer the other.

In general, TraumAID 2.0 plans were preferable for being methodical, organized and avoiding errors of omission. In at least two cases, it is believed that the patient's death could have been prevented had TraumAID 2.0's management been followed:

- Case 900425 – This 16 year old male presented unstable with a gunshot wound to the back. The actual care was criticized by the judges for *missing* a heart injury; not aspirating the chest before inserting a chest tube; and doing an unnecessary IVP. The patient died as a result of recurrent shock. TraumAID 1.0 was criticized for not taking an abdominal film even though the bullet could not be located on the chest X-ray. TraumAID 2.0 was judged perfectly acceptable by two of the judges and NME by the third. The judges thought that the

Actual Care \ TraumAID 2.0	ACC	NME	RES	UNACC	Total
ACC	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 0	Actual Care: 1 | 2 | 3 — *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 1	Actual Care: 1 | 2 | 3 — *1* *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 2	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 0	Actual Care: 1 | 2 | 3 — *1* *1* *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 3
NME	Actual Care: 1 | 2 | 3 — ; Neutral: 1 ; TraumAID 2: 1 | 2 | 3 — *3* *3* *1* ; 8	Actual Care: 1 | 2 | 3 — *1* ; Neutral: 8 ; TraumAID 2: 1 | 2 | 3 — *5* *7* *1* ; 22	Actual Care: 1 | 2 | 3 — *4* *2* *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 7	Actual Care: 1 | 2 | 3 — *1* *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 2	Actual Care: 1 | 2 | 3 — *6* *2* *2* ; Neutral: 9 ; TraumAID 2: 1 | 2 | 3 — *8* *10* *2* ; 39
RES	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — *1* *3* *1* ; 5	Actual Care: 1 | 2 | 3 — *1* ; Neutral: 5 ; TraumAID 2: 1 | 2 | 3 — *2* *12* *5* ; 25	Actual Care: 1 | 2 | 3 — *2* ; Neutral: 1 ; TraumAID 2: 1 | 2 | 3 — *6* *1* ; 10	Actual Care: 1 | 2 | 3 — *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — ; 1	Actual Care: 1 | 2 | 3 — *3* *1* ; Neutral: 6 ; TraumAID 2: 1 | 2 | 3 — *9* *16* *6* ; 41
UNACC	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — *4* ; 4	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — *3* *2* ; 5	Actual Care: 1 | 2 | 3 — ; Neutral: 1 ; TraumAID 2: 1 | 2 | 3 — *1* *2* ; 4	Actual Care: 1 | 2 | 3 — ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — *1* ; 1	Actual Care: 1 | 2 | 3 — ; Neutral: 1 ; TraumAID 2: 1 | 2 | 3 — *2* *3* *8* ; 14
Total	Actual Care: 1 | 2 | 3 — ; Neutral: 1 ; TraumAID 2: 1 | 2 | 3 — *4* *6* *6* ; 17	Actual Care: 1 | 2 | 3 — *3* ; Neutral: 13 ; TraumAID 2: 1 | 2 | 3 — *7* *22* *8* ; 53	Actual Care: 1 | 2 | 3 — *6* *3* *2* ; Neutral: 2 ; TraumAID 2: 1 | 2 | 3 — *7* *1* *2* ; 23	Actual Care: 1 | 2 | 3 — *1* *1* *1* ; Neutral: 0 ; TraumAID 2: 1 | 2 | 3 — *1* ; 4	Actual Care: 1 | 2 | 3 — *10* *4* *3* ; Neutral: 16 ; TraumAID 2: 1 | 2 | 3 — *19* *29* *16* ; 97

Actual Care

Figure 6.17: Refined Comparison of TraumAID 2.0 and Actual Care

patient's death resulted from errors in the actual management.

- Case 900024 – This 30 year old female arrived at the emergency room unstable with a stab wound to the back. The actual care missed a heart injury and a persistent hemothorax. The patient died as a result of recurrent shock. TraumAID 2.0 was not ranked perfectly acceptable on this case due to disagreement between the judges on whether or not to insert a chest tube in a patient who is about to undergo a chest surgery on the same side of the chest.

While in some cases, errors of commission that were avoided by TraumAID 2.0 were praised, there were a few cases in which TraumAID 2.0 was criticized for *not* pursuing a diagnosis. Following is an examination of cases in which TraumAID 2.0 ranked unacceptable and the actual care was acceptable at various levels:

- Case 900495 – This patient had two gunshot wounds to the abdomen, one pointing up and the other down (in retrospect, this was a through-and-through wound, i.e. a pair of entry and exit wounds). TraumAID 2.0 was criticized by the judges for having the patient rushed to the operating room for a laparotomy without evaluating the chest first. TraumAID 1.0 *did* order an abdominal X-ray, which was found normal, and the eventual treatment was the same. In the domain expert's opinion, TraumAID 2.0's management was in fact better since further examination couldn't have changed the recommended treatment. TraumAID 2.0 may have outsmarted itself, and the judges, in this case.

- Case 900898 – In this case, too, the patient suffered a gunshot wound to the abdomen, and TraumAID 2.0 was criticized for not taking a *chest* X-ray before rushing the patient to the operating room. Here too, the domain expert believes the management was correct. The patient suffered a simple hemothorax on both sides, which was also missed in the actual care. The chest X-ray was negative and it is believed that the hemothoraces developed post-operatively.

- Case 900527 – The criticism in this case is very similar to the above two cases. The patient was shot in the abdomen. After discovering the bullet in the abdomen, TraumAID 2.0 called for a laparotomy. Two of the judges felt that a *chest* X-ray should also have been taken (in the actual management, a chest X-ray came out normal). The domain expert believes that this is mostly a matter of routine and that TraumAID's recommendation was rational. The third judge ranked *all* three managements unacceptable for lack of more complete abdominal evaluation.

To summarize, in all cases in which TraumAID 2.0's recommendations were judged unacceptable, it was due to differences in opinion between the domain expert on whose practices the rules are based and those of the judges and not due to a particular weakness of TraumAID 2.0's methodology.

Figure 6.18 presents a refined comparison of the management plans of TraumAID 2.0 and TraumAID 1.0.

There were only three cases in which TraumAID 1.0 was judged preferable to TraumAID 2.0 by at least one category difference:

- Case 900088 – This case involves a patient with two stab wounds to the chest, which was eventually only put on a periodic evaluation. There was little difference in the relative evaluation of TraumAID 1.0, TraumAID 2.0 and the actual care. However, and despite the lack of complications, there was a serious disagreement between the judges about the general level of care provided. In particular, one physician rated all three plans unacceptable (for lack of sufficient objective abdominal evaluation) whereas another rated all three acceptable with no major errors. TraumAID 1.0's plan was slightly preferred over TraumAID 2.0's by the third physician (ACC vs. NME) for taking a chest X-ray *before* covering the wounds. The third physician rated the actual plan unacceptable.

TraumAID 1.0 \ TraumAID 2.0	ACC	NME	RES	UNACC	Total
ACC	TraumAID 1: 1 | 2 | 3 — Neutral: 2 — TraumAID 2: 1 | 2 | 3; *1* — 3	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 0	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 0	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 0	TraumAID 1: 1 | 2 | 3 — Neutral: 2 — TraumAID 2: 1 | 2 | 3; *1* — 3
NME	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3; *1* | *2* | *1* — 4	TraumAID 1: 1 | 2 | 3; *1* | *2* — Neutral: 12 — TraumAID 2: 1 | 2 | 3; *4* | *13* | *1* — 33	TraumAID 1: 1 | 2 | 3; *1* — Neutral: 0 — TraumAID 2: 1 | 2 | 3; *1* — 2	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 0	TraumAID 1: 1 | 2 | 3; *2* | *2* — Neutral: 12 — TraumAID 2: 1 | 2 | 3; *6* | *15* | *2* — 39
RES	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3; *5* | *4* — 9	TraumAID 1: 1 | 2 | 3 — Neutral: 1 — TraumAID 2: 1 | 2 | 3; *3* | *15* | *1* — 20	TraumAID 1: 1 | 2 | 3; *1* | *1* — Neutral: 9 — TraumAID 2: 1 | 2 | 3; *5* | *4* | *0* — 20	TraumAID 1: 1 | 2 | 3; *1* — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 1	TraumAID 1: 1 | 2 | 3; *1* | *2* — Neutral: 10 — TraumAID 2: 1 | 2 | 3; *8* | *24* | *5* — 50
UNACC	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3; *1* — 1	TraumAID 1: 1 | 2 | 3 — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 0	TraumAID 1: 1 | 2 | 3; *1* — Neutral: 0 — TraumAID 2: 1 | 2 | 3 — 1	TraumAID 1: 1 | 2 | 3; *1* — Neutral: 2 — TraumAID 2: 1 | 2 | 3 — 3	TraumAID 1: 1 | 2 | 3; *2* — Neutral: 2 — TraumAID 2: 1 | 2 | 3; *1* — 5
Total	TraumAID 1: 1 | 2 | 3 — Neutral: 2 — TraumAID 2: 1 | 2 | 3; *2* | *7* | *6* — 17	TraumAID 1: 1 | 2 | 3; *1* | *2* — Neutral: 13 — TraumAID 2: 1 | 2 | 3; *7* | *28* | *2* — 53	TraumAID 1: 1 | 2 | 3; *3* | *1* — Neutral: 9 — TraumAID 2: 1 | 2 | 3; *6* | *4* — 23	TraumAID 1: 1 | 2 | 3; *1* | *1* — Neutral: 2 — TraumAID 2: 1 | 2 | 3 — 4	TraumAID 1: 1 | 2 | 3; *5* | *4* — Neutral: 26 — TraumAID 2: 1 | 2 | 3; *15* | *39* | *8* — 97

Figure 6.18: Refined Comparison of TraumAID 2.0 and TraumAID 1.0

- Case 900024 – While two of the physicians ranked both TraumAID 1.0 and TraumAID 2.0 as NME and indicated slight preference for TraumAID 2.0, the third physician did not like TraumAID 2.0 management and rated it as UNACC. The disagreement was over whether to put a chest tube in a patient suffering a hemothorax who is due to undergo surgery, or to skip it and rush the patient to the operating room. TraumAID 2.0's planner will normally suggest a chest tube insertion in patients suffering a hemothorax. However, consistent with the opinion of the domain expert, a special optimizer was added to override this recommendation in patients about to undergo a chest operation. This optimization is also consistent with the opinion of two of the judges. As an aside, the actual care was rated UNACC by all three judges.

- Case 900495 – see above.

To conclude, TraumAID 2.0 plans were judged to be significantly more acceptable than either the actual care, or the plans proposed by TraumAID 1.0. It is also interesting to note that among the case managements that had to be sent back to the judges for reconsideration (cases in which there was no consensus among the judges), 33 were actual care, 41 were TraumAID 1.0 cases, and only 20 were TraumAID 2.0 cases. Thus, the judges were more consistent in their opinion about TraumAID 2.0's plans. As a final note, some of the cases were actually managed by members of the panel, although except in one instance specific cases were not recognized as such. Thus, the surgeons often *preferred TraumAID 2.0's management to same of their own.*

6.3 Further Comparison of TraumAID 2.0 and TraumAID 1.0

The first version of TraumAID, TraumAID 1.0, was a *rule-based* consultation system for multiple trauma management. Like TraumAID 2.0, its scope includes gunshot and stab wounds to the chest and abdomen. Because rules were found to be inadequate as a single representation and reasoning vehicle, TraumAID 2.0's objective was to augment TraumAID 1.0 with more principled reasoning about *actions*.

The main difference between TraumAID 1.0 and TraumAID 2.0 is that while in the former one has to explicitly encode diagnoses and responses to any *combination* of injuries that can be sustained by a patient, the latter's ECM architecture allows *implicit* coordination of *local* strategies.

Besides its new planner, this required modifying TraumAID 1.0's original rules to be goal-directed so that they *effectively* post goals for the planner. Before, questions and actions were presented to the physician in a pre-defined, fixed order. Questions and diagnostic tests were collected, in a one-by-one fashion, throughout the interactive session. Therapeutic actions, excluding those that are both diagnostic and therapeutic, were accumulated in a *prescription* list, also based on a fixed order. In TraumAID 2.0, the planner addresses *combinations* of goals. *Complete* plans are presented, and adapted throughout the interactive session. These plans intersperse diagnosis and therapy. Various diagnostic and therapeutic needs are coordinated in a principled way by an implicit planning algorithm. Among the principles used in TraumAID 2.0's planner are logistical considerations which were not at all present in TraumAID 1.0. Finally, some other weaknesses of TraumAID 1.0 were also improved upon in TraumAID 2.0.

The redesign of TraumAID's rules, particularly avoiding the need to explicitly reason about many combinations of injuries, have simplified them significantly. The number of antecedents per rule – an indicator of its complexity – has been reduced from an

average of 2.80 in TraumAID 1.0 to an average of 2.32 in TraumAID 2.0's evidential rules and 1.69 in its goal-setting rules, even though TraumAID 2.0's domain is somewhat *more* complex.

This simplification is, to a great extent, due to the fact that TraumAID 1.0 needed to have complex rules to get around the fixed order of actions in its management plans, when the particular configuration of injuries needed to be treated differently. A second simplification in the rules came from the use of *inhibition clauses*, implemented in TraumAID 2.0 as a so-called *hierarchy of goals*, which are *automatically* compiled into the rules when they are loaded. Many of the goal-setting rules replaced so-called suspect rules in TraumAID 1.0 which were used to control information gathering. Other goal-setting rules were added for previously unhandled therapeutic goals. TraumAID 2.0 uses 111 goal-procedure mappings and 79 procedure-action mappings.

While developed, TraumAID 2.0 was validated using a collection of 270 *theoretical* trauma cases which were hand-crafted by our domain expert during the development of TraumAID 1.0 for its own validation. Following are a few cases from that collection which exemplify some of the new features of TraumAID 2.0:

Case tp6-302b.

In this patient who was shot in the left lower abdomen, TraumAID 2.0 decided *not* to pursue an arteriogram after the need for a laparotomy was established by the identification of a bullet in the abdominal film. The mechanism used for that purpose is a goal-inhibition rule.

Case tp6-317c.

This patient was stabbed in the right lower abdomen. He was not in shock, and was not obtunded. There were no adverse clinical signs (ileus, guarding, rebound tenderness), and the pulses were normal. Blood in the urine, however, triggered TraumAID 1.0 to request a series of tests: a cystogram, an IVP, and a peritoneal lavage. TraumAID 2.0, given the stable condition of the patient, recommended

replacement of the IVP and lavage with a single CT scan. The mechanism used for that purpose is the planner's parsimonious set-covering model. In any case, the results were negative and the patient was observed.

Case tp6-120g.

This patient was unstable after being stabbed in the upper abdomen (epigastrium). The neck veins were not distended and there were no muffled heart sounds. After the chest X-ray was also negative, TraumAID 2.0 has focused on abdominal bleeding as the cause of shock, and recommended proceeding with a lavage. TraumAID 1.0 first called for a pericardiacentesis, which was negative; then it called for a local wound exploration, which was obviously positive; and only then recommended the lavage.

This case demonstrates a combination of possible injuries which was simply not well covered by TraumAID 1.0. While normally attributed to chest wounds, here one had to reason about the potential of a heart injury given an abdominal wound. TraumAID 1.0 missed the lack of clinical signs and proceeded to aspire the pericardial sac. In TraumAID 2.0, there is little need to encode the interaction and so the heart injury hypothesis was simply ruled out on the basis of the regular clinical examination. TraumAID 2.0 thus proceeded immediately to the abdomen. In the abdomen, given that the patient was in shock, the outcome of the local wound exploration was irrelevant to the continued treatment. TraumAID 2.0 was thus instrcted to skip over it, using a goal-inhibition mechanism, and proceeded directly to lavage the patient.

Case tp6-114g.

This patient was stabbed in the lower left abdomen. Compared to TraumAID 1.0, TraumAID 2.0 did not ask for a local wound exploration, and replaced an IVP and lavage with a single CT scan.

Case tp56-303b.

This patient had two gunshot wounds on opposite sides of the lower chest. TraumAID 1.0 called for the diagnosis and treatment of a simple pneumothorax on both sides, but did not pursue a possible abdominal injury. Since none of the bullets were located, it is possible that both headed to the abdomen. Even if it were a through-and-through wound, i.e. one of the wounds was an exit wound, it is still possible that the abdomen was lacerated given that the wounds were very low in the chest. TraumAID 2.0 would have called for interspersing diagnosis and, if necessary, treatment of the abdomen, with the treatment of the pneumothoraces.

Case tp56-220b.

This case also illustrates improved completeness of the evaluation. The patient was stabbed in the lower left chest, and unstable. TraumAID 1.0 asked for clinical examination of the chest and left the patient untreated when it was all negative (although no X-ray was taken!). TraumAID 2.0 recommended investigating both the chest and the abdomen. Given the negative clinical examination of the chest, it first called for looking at abdominal bleeding as a possible cause of shock. (Given the urgency, it chose a lavage for that purpose.) When the lavage was negative it shifted its focus to a pericardial tamponade as an alternative cause of shock.

Case tp56-206.

This patient was shot in the upper left chest and in the abdomen, and was unstable. A chest X-ray showed a single bullet in the chest and also a hemothorax. The hemothorax persisted even though it was treated with a chest tube. TraumAID 2.0 requested an abdominal X-ray, to locate the other bullet, before transferring the patient to the operating room. TraumAID 1.0 requested, in addition, an arteriogram (thoracic), a barium swallow study, and a bronchoscopy. The mechanisms used to prevent those are goal inhibition clauses and time constraints (arteriogram and bronchoscopy are too lengthy for an unstable patient).

Cases tp56-205, 204d.

In these cases, one patient was shot in the upper abdomen, the other was shot twice in the chest and sustained an abdominal injury. In both cases, TraumAID 2.0 recommended overriding the need for an abdominal arteriogram given the already established need for a laparotomy. The mechanism used was goal inhibition.

Case tp56-181.

This patient was unstable, after being shot in the upper abdomen, and was diagnosed as suffering a pericardial tamponade. After an X-ray showed a left pneumothorax, TraumAID 1.0 suggested decompression with a chest tube. TraumAID 2.0 suggested skipping the chest tube insertion, and instead rushing the patient to the operating room for a planned bilateral surgery. While preferable in the opinion of the domain expert, this practice was occasionally criticized by one of the judges who prefers to go ahead with the chest tube insertion (see Section 6.2). In TraumAID 2.0, the prophylaxis is suppressed by one of the domain specific optimizers.

6.4 Summary

I have presented TraumAID 2.0 – an implemented consultation system for the trauma domain. TraumAID 2.0 implements an ECM architecture with a GDD reasoner and a planner as described in Sections 3.9 and 4.3 respectively. JR's case, which was used earlier to demonstrate important features of the multiple trauma management domain, is also used here as an example. I then presented an empirical study in which mangement produced by TraumAID 2.0 for real trauma patients was compared to the actual care administered and to management proposed by TraumAID 1.0. In the judgement of three trauma experts, TraumAID 2.0's plans were significantly preferable (by ratios of 64:17 and 62:9 respectively). Cases in which TraumAID 2.0's plans were judged to be *weak* are analyzed and explained. I also present cases which demonstrate the improvements of TraumAID 2.0 over its predecessor TraumAID 1.0.

Chapter 7

A Catalogue of Diagnosis-and-Repair Strategies

7.1 Overview

The work reported in this dissertation started as an engineering project. Only after TraumAID 2.0 was already up and running have I tried to identify the principles used in its encoding and to formulate a theory that would account for these principles. To close the loop, it was necessary to show that the new formalization is indeed (a) *useful* and (b) *natural*.

Considering that the GDD implementation of TraumAID 2.0 is different from the formulation of Chapter 3, the first step involved implementing the new formulation and testing it on pieces of TraumAID 2.0's knowledge that were transformed to the new representation (cf. Section 3.7). Having done that, I embarked on a larger study in which I tried to identify important diagnostic and therapeutic strategies. In particular, I was searching for

1. strategies that commonly occur in diagnosis and/or diagnosis-and-repair domains;

2. strategy-segments that commonly serve as "building blocks" for such strategies; and

3. strategies that represent important diagnosis-and-repair "philosophies".

The original purpose of this study was to show that these strategies *can* be encoded in the ECM framework, and that such encoding is natural, i.e. that these strategies are inherently goal-directed. From an engineering perspective, identifying such strategies facilitates the creation of strategy *templates* for future use. From a scientific perspective, this study has retrospectively resulted in an intriguing collection of diagnosis-and-repair strategies.

Scientists have studied diagnostic and therapeutic strategies for a variety of reasons. Many have studied medical decision making with the purpose of educating physicians [114, 42, 79, 39, 81]. Others, [96, 5, 23], have studied such strategies with the objective of eventually teaching them to a machine. Yet others have used them as *insights* on how to encode medical knowledge in computerized decision support systems, e.g. the systems in [147, 22].

The notion of strategy is clearly useful, and is sometimes explicitly present in medical management programs, e.g. the skeletal plans of ONCOCIN [70, 92] and ATTENDING [104]. However, in my search, I found that diagnostic strategies are much better studied and documented than ones in which diagnosis and therapy are *mixed*. This is particularly true of references to *abstract* strategies. In Section 7.2, I present a collection of primitive *building blocks* for abstract diagnostic strategies which I have collected from the literature. I discuss the GDD encoding for those, and exemplify them in two composite strategies.

I have been more interested in strategies that *mix* diagnosis and repair, which I present in Section 7.3. Although most of these strategies were first identified in the trauma domain (the remainder being pointed out to me by colleagues working in other domains), I believe that the principles demonstrated in them are of general

importance to decision making in exploratory-corrective domains, and maybe other domains as well. I found this catalogue of strategies, although somewhat murky and definitely incomplete, to be intriguing, and I further believe it should be of interest to other workers in medical decision making and Artificial Intelligence in general.

7.2 Building Blocks for Diagnostic Strategies

7.2.1 Hypothesis Generation

Throughout the course of management, and as early as the very first bits of information become available, a diagnostician often *hypothesizes* which disease the patient may suffer, and then works to confirm, or refute, his hypotheses. This process was observed in clinicians by Kassirer and Gorry [79]:

> "Requesting and assessing new information, they rejected some of the initial hypotheses, substituted specific hypotheses for more general ones, and selected a few specific hypotheses for detailed and critical testing".

Similar observations were made by other researchers [42]. The set of current hypotheses is often referred to as the *working hypotheses*.

In the end, none of the hypothesized diseases may be present, but the role of *exhaustive* hypothesis generation is still important, as emphasized by Clendening and Hashinger [25]: "the most brilliant diagnosticians of my acquaintance are the ones who do remember and consider the most possibilities". Hilliard [71] notes that

1. In contrast to the way students are often taught, experienced physicians generate their hypotheses early on;

2. A very limited number of hypotheses are considered at one time; five to seven at most. They are then eliminated or retained depending on new information;

3. The most common error in the confirmation/elimination process is that of *overinterpretation*. That is inappropriately attributing new information to a current hypothesis, instead of generating a new one.

In general, it seems that knowledge-based systems can improve this process. Once programmed, it is easier for a machine to produce and consider an exhaustive set of hypotheses. A machine is also less prone to human weaknesses such as limited ability to consider more than a few hypotheses [102]; premature closure [81]; and poor ability to reason with probabilities [77]. In GDD, goal-setting rules can be used to trigger new hypotheses in the form of "relevant" facts. For example, in JR's case, the chest wound triggered the initial investigation of a possibility of tension pneumothorax:

Chest_Wound (Side=S)
$\triangleright$ *Possibility_of_Tension_Pneumothorax(Side=S)*

7.2.2 Generalization and Specialization

Two basic steps in diagnostic strategies are *generalization* and *specialization*. Given a disease, a *generalization* strategy can be used to generate a more general class of diseases to which that disease belongs. This is useful because there is often more knowledge associated with more general classes, which might become useful in the continued diagnosis and treatment of the current disease. In addition, in the presence of uncertainty, other members of the general class should sometimes be considered as alternative diagnoses given that they often have similar symptoms. *Specialization* is the opposite process: a general class of diseases is specialized, or refined. Specialization is important where there are differences in the treatment of different sub-classes. If a general class is identified, more information may be needed in order to specialize. This may require more tests.

Figure 7.1 presents an example used by Alpay [5] to demonstrate generalization and specialization. It deals with diagnosing orthopedic back pain; the flow indicates

specialization of the diagnosis. Generalization works in the opposite direction.

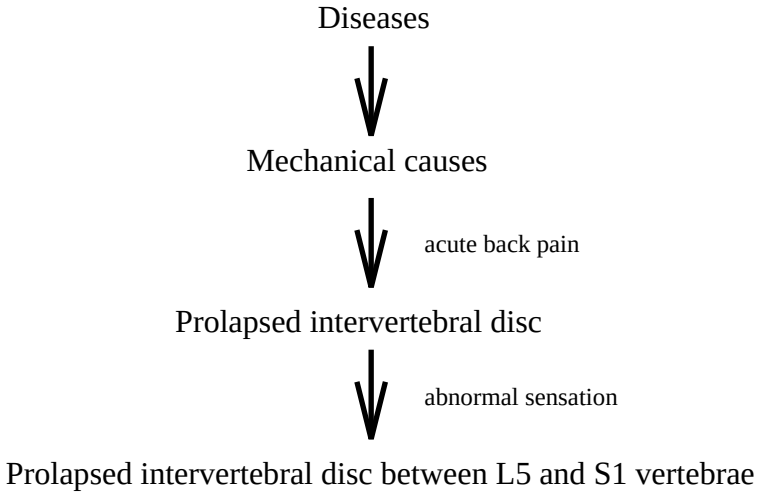


Figure 7.1: Orthopedic Back Pain Diagnosis

Alpay discusses three types of errors that are common in generalizing/specializing: (a) *incorrect* generalization/specialization steps, (b) *overgeneralization* or *overspecialization*, and (c) *undergeneralization* or *underspecialization*.

GDD-style encoding can help reduce errors of the latter two types. The basic GDD encoding of a generalization strategy for this example is:

Prolapsed_intervertebral_disc_between_L5_and_S1_vertebrae
$\Rightarrow$ *Prolapsed_intervertebral_disc*

Prolapsed_intervertebral_disc
$\Rightarrow$ *Mechanical_causes*

Mechanical_causes
$\Rightarrow$ *Diseases*

Overgeneralization occurs when a too general conclusion is made. By the GDD principle, no generalization is too general unless it may lead to unnecessary or incorrect *action*. Thus, if an overgeneralization step can lead to an unwarranted investigation, then it is *that investigation* that should be inhibited, not the generalized *conclusion*. For example, if the general *Diseases* conclusion leads to testing the possibility of

either inflammatory or mechanical causes, as in the following rules:

$$Diseases \triangleright Mechanical\_causes$$

$$Diseases \triangleright Inflammatory\_causes$$

and if further the two are *known* to be exclusive, then there is no need to pursue the inflammatory causes path. To prevent that from happening, the above rules can be augmented as follows:

$$Diseases \wedge unless(Inflammatory\_causes) \triangleright Mechanical\_causes$$

$$Diseases \wedge unless(Mechanical\_causes) \triangleright Inflammatory\_causes$$

Thus, the *Diseases* conclusion will remain while the unnecessary investigation will be curbed. Alternatively, a goal-inhibition rule can be used, taking advantage of the fact that the two specializations are mutually exclusive:

$$Diseases \wedge Inflammatory\_causes \Rightarrow \neg Mechanical\_causes$$

$$Diseases \wedge Mechanical\_causes \Rightarrow \neg Inflammatory\_causes$$

Undergeneralization will never occur if all generalization rules are explicitly written, as above.

Specialization[1] works in the opposite direction and may require additional information. In the above example, if the problem is believed to be of a mechanical nature,

[1]The specialization strategy is also referred to as *refinement*, e.g. [113].

then the following rules can be used to further specialize:

$$\begin{aligned} &\mathit{Mechanical\_causes} \wedge \\ &\mathit{Acute\_back\_pain} \\ &\qquad \Rightarrow \mathit{Prolapsed\_intervertebral\_disc} \end{aligned}$$

$$\begin{aligned} &\mathit{Prolapsed\_intervertebral\_disc} \wedge \\ &\mathit{Abnormal\_Sensation} \\ &\qquad \Rightarrow \mathit{Prolapsed\_intervertebral\_disc\_between\_L5\_and\_S1\_vertebrae} \end{aligned}$$

For these rules to be applicable, additional information needs be gathered. The following goal-setting rules can be used to let the planner know such specializations need be investigated. The planner will, in turn, call for action to make such information available.

$$\begin{aligned} &\mathit{Mechanical\_causes} \\ &\qquad \triangleright \mathit{Prolapsed\_intervertebral\_disc} \end{aligned}$$

$$\begin{aligned} &\mathit{Prolapsed\_intervertebral\_disc} \\ &\qquad \triangleright \mathit{Prolapsed\_intervertebral\_disc\_between\_L5\_and\_S1\_vertebrae} \end{aligned}$$

Alpay defines *overspecialization* as specialization that is not supported by available data. This particular form of overspecialization can be trivially avoided if the specialization rule is qualified by that data. A more interesting case of overspecialization not considered by Alpay is where diagnostic *activity* is *wasted* in gathering information for the purpose of unnecessary specialization. For example, if patients over the age of 75 cannot safely be treated with surgery and if otherwise the treatment for the particular *Prolapsed intervertebral disc between L5 and S1 vertebrae* disease is the same as for the more general disease *Prolapsed intervertebral disc*, then gathering evidence of abnormal sensation is unnecessary. This can be prevented using the following rule:

$$\begin{aligned} &\mathit{Age\_is\_over\_75} \\ &\qquad \triangleright \neg \mathit{Prolapsed\_intervertebral\_disc\_between\_L5\_and\_S1\_vertebrae} \end{aligned}$$

Since a condition may often be concluded in a variety of ways, a specialized condition may be concluded before its generalizations are. If that happens, we would often like to prevent the latter from being *actively* pursued, even if they become relevant in some later point in time. In the previously presented case, the generalization rules will be effective in preventing such activity: more general diseases are concluded positive and labeled *relevant* (attitude=R), but *known* (belief=T or F). An alternative goal-inhibition approach is presented when scaled diagnosis is discussed, where a similar situation occurs.

As with undergeneralization, *underspecialization* will never occur if all specialization rules are specified.

As presented so far, the *specialization* strategy is driven by current hypotheses. Specialization can also be symptom-driven[2]. The application of a symptom-driven specialization is often based on the physician's own experience and/or on medical protocols. Given an observation, specialization is recommended, but *not* with a particular hypothesis in mind. For example, a patient complaining about lower back pain will often be asked about the nature of the pain, i.e. whether it was acute (i.e. occurred suddenly) or chronic. While there is probably a good, "deep" justification for collecting such data (e.g. it may have been found useful in constraining the range of considered diseases, the specialization step is better characterized (i.e. simpler, more intuitive) as being primarily driven by the first observation. In TraumAID, for example, a chest X-ray is *always* requested for any patient with a gunshot or stab wound to the chest. Thus, rather than enumerating all possible reasons (suspected diseases), it is probably simpler to write a rule of the form:

Chest_Wound (Side=S)
$\triangleright$ *Survey_Chest_X-Ray*

[2] Alpay [5] separates such strategies as *problem refinement* strategies, but that terminology may be confusing given that refinement was previously referring to any specialization.

The *explore* strategy [113] can be viewed as a special case of a symptom-based specialization strategy. In that strategy, more information is gathered when there is *not enough* information in the patient's case to generate hypotheses.

7.2.3 Confirmation and Elimination

During the course of diagnosis, a variety of competing hypotheses can be generated. Hypotheses are often competing in that the presence of one rules out, of reduces the chance of the presence of others. Similarly, the absence of one can confirm, or raise the odds of others. Two types of steps are commonly taken to narrow down the set of working hypotheses: *confirmation* and *elimination*. Confirmation uses data to *increase* the perceived likelihood of a diagnosis whereas elimination uses data to *decrease* the likelihood of other hypotheses. *Discrimination*, sometimes taken as a third type, can often be viewed as a combination of elimination and confirmation. An hypothesis can be confirmed via *sufficient*, or *necessary positive* evidence. Likewise, it can be eliminated via *negative* evidence or via the *absence of necessary evidence*. In addition, hypotheses are sometimes eliminated due to *insufficient evidence*. (Note that the latter reflects a negative bias).

The use of confirmation and elimination is demonstrated next using *composite* strategies.

7.2.4 Examples

Scaled Diagnosis

Scaled diagnosis is a strategy in which a *sequence* of steps is taken, until a diagnosis is finally reached. The order in which steps are taken will often correspond to their cost or risk to the patient. (as aside, it is often the case that more costly/risky steps

are also more accurate predictors.) Thus, such a strategy may be used to *screen* patients by first using tests which are cheap, and less risky, but with possibly high false positive rates, before using more definitive tests. Steps requiring information available via general physical examination, which is typically performed on every patient, will usually appear earlier in a scaled diagnosis strategy for a given problem. Scaled diagnosis is frequently used in trauma management, where it involves proceeding from initial symptoms through a number of *cautious* information acquisition steps until a hypothesis is either confirmed or ruled out.

Consider, for example, a patient with a gunshot wound to the chest which is investigated as possibly having a tension pneumothorax (this example was previously used in Section 3.7 to illustrate the use of GDD rules). An hypothesis generation step, modeled via a goal-setting rule, may call for the investigation of the *possibility of* tension pneumothorax:

Chest_Wound (Side=S)
$\triangleright$ *Possibility_of_Tension_Pneumothorax(Side=S)*

In response, an elimination strategy will be used to try to rule out that possibility. Tests of increasing cost and risk, but also of increasing validity, will be taken in sequence. If any of them fails, then the patient is determined *not* to have a tension pneumothorax. Figure 7.2 portrays that process graphically.

First, given the initial hypothesis, information regarding *shock* and *distended neck veins* will be called for. In turn, that information will be used to conclude positively:

Chest_Wound (Side=S) $\wedge$
Shock
$\Rightarrow$ *Possibility_of_Tension_Pneumothorax(Side=S)*

Chest_Wound (Side=S) $\wedge$
Distended_neck_veins
$\Rightarrow$ *Possibility_of_Tension_Pneumothorax(Side=S)*

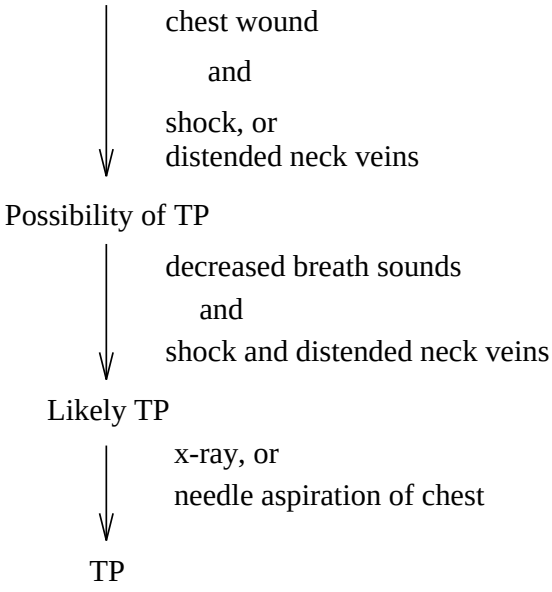


Figure 7.2: Scaled Diagnosis – Diagnosing Tension Pneumothorax

and to proceed to the next step in the strategy:

Possibility_of_Tension_Pneumothorax(Side=S)
$\triangleright$ *Likely_Tension_Pneumothorax(Side=S)*

Note that information and conclusions may be acquired and made asynchronously to the particular diagnostic path, e.g. through concurrently pursued diagnostic and therapeutic strategies. Thus, the patient may be known to be in shock at the time when the *Possibility_of_Tension_Pneumothorax* goal was first set. In such case, the reasoner would have already been able to conclude the first step positively *without asking* whether or not the patient suffers from distended neck veins. This demonstrates confirmation via sufficient information.

To proceed with the strategy, information concerned with the patient's ability to

breathe is required:

$$\begin{aligned}
&\textit{Possibility_of_Tension_Pneumothorax(Side=S)} \wedge \\
&\textit{Decreased_breath_sounds (Side=S)} \\
&\qquad \Rightarrow \textit{Likely_Tension_Pneumothorax(Side=S)}
\end{aligned}$$

$$\begin{aligned}
&\textit{Likely_Tension_Pneumothorax(Side=S)} \\
&\qquad \rhd \textit{Tension_Pneumothorax(Side=S)}
\end{aligned}$$

In the final step of the strategy, one of two tests need to be performed in order to confirm or eliminate the *Tension_Pneumothorax* diagnosis. Either an X-ray or a needle aspiration will be chosen, depending on the patient's state.

$$\begin{aligned}
&\textit{X-ray_shows_Tension_Pneumothorax(Side=S)} \\
&\qquad \Rightarrow \textit{Tension_Pneumothorax(Side=S)}
\end{aligned}$$

$$\begin{aligned}
&\textit{Needle_aspiration_shows_pressure_in_chest(Side=S)} \\
&\qquad \Rightarrow \textit{Tension_Pneumothorax(Side=S)}
\end{aligned}$$

In practice, confirmation/elimination need not always proceed sequentially within the scaled diagnosis strategy. Suppose, for example, that an X-ray was taken to evaluate another injury and has shown a tension pneumothorax. By the GDD principle, it would be wasteful to go after the decreased breath sounds sign, as it can only contribute to an already determined diagnosis. To prevent such behavior one can further qualify the goal-setting rules:

$$\begin{aligned}
&\textit{Chest_Wound (Side=S)} \wedge \\
&\textit{unless(Likely_Tension_Pneumothorax(Side=S))} \wedge \\
&\textit{unless(Tension_Pneumothorax(Side=S))} \\
&\qquad \rhd \textit{Possibility_of_Tension_Pneumothorax(Side=S)}
\end{aligned}$$

$$\begin{aligned}
&\textit{Possibility_of_Likely_Tension_Pneumothorax(Side=S)} \wedge \\
&\textit{unless(Tension_Pneumothorax(Side=S))} \\
&\qquad \rhd \textit{Likely_Tension_Pneumothorax(Side=S)}
\end{aligned}$$

or alternatively, one can impose a generalization strategy on top of the scaled diagnosis strategy:

Likely_Tension_Pneumothorax(Side=S)
$\Rightarrow$ *Possibility_of_Tension_Pneumothorax(Side=S)*

Tension_Pneumothorax(Side=S)
$\Rightarrow$ *Likely_Tension_Pneumothorax(Side=S)*

That way, even though earlier goals in the sequence will remain labeled "relevant", their underlying concepts will be tagged known and thus not pursued actively.

Differential Diagnosis

Differential diagnosis (also referred to as discrimination strategy) also involves hypothesis generation, elimination, and confirmation. It has been used in several computer systems, e.g. INTERNIST-I [103, 123] (internal medicine), ABEL [112] (electrolyte and acid-based disturbances), and PATHFINDER [73] (pathological lymph node tissue analysis.)

Typically, the process begins with one or more hypothesis generation steps in which the physician lists all diseases which the specific case can reasonably resemble. The purpose of the process is then to eliminate all but one of the hypothesized diseases, which is then confirmed. Tests are used to differentiate among the multiple hypotheses, and a good strategy is one which is most efficient in the tests it uses. New hypotheses can sometimes be generated, depending on the outcome of test results. Figure 7.3 illustrates graphically a typical differential diagnosis strategy.

Given the initial evidence e, four competing hypotheses are generated $d_1, \cdots, d_4$ (and let us assume that only one of the 4 diseases can be present). The first test t_1 can be used to differentiate the first two from the last, e.g. d_3, and d_4 are ruled out

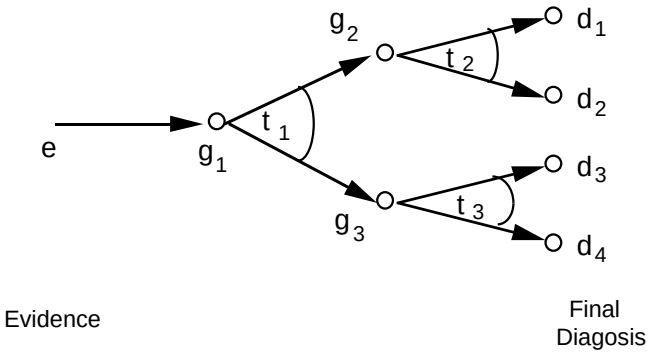


Figure 7.3: Differential Diagnosis

if t_1 comes back positive, and otherwise d_1, and d_2 are eliminated. Depending on the remaining hypotheses, another test (t_2 or t_3) is selected to pinpoint the actual disease. In GDD, this strategy is captured by the following set of goal-setting rules:

$$e \wedge t_1 \wedge t_2 \Rightarrow d_1$$

$$e \wedge t_1 \ \triangleright g_2$$

$$e \wedge t_1 \wedge \neg t_2 \Rightarrow d_2$$

$$e \ \triangleright g_1$$

$$e \wedge \neg t_1 \wedge t_3 \Rightarrow d_3$$

$$e \wedge \neg t_1 \ \triangleright g_3$$

$$e \wedge \neg t_1 \wedge \neg t_3 \Rightarrow d_4$$

Again, the problem is that there are often a number of methods to conclude various diseases. In particular, it is possible that in a given scenario, d_4 will be concludable via some other inference mechanism. In such occasion, if the above rules are used alone, the tests t_1 and subsequently t_3 will be pursued – clearly not a desirable feature. The following goal inhibition rules can be used to curb unnecessary tests in such instance:

$$d_4 \ \triangleright \neg g_1 \qquad d_4 \ \triangleright \neg g_2 \qquad d_4 \ \triangleright \neg g_3$$

Even if a diagnosis was not yet made, its treatment may have already been recommended for other purposes. Suppose, for example, that t_1 came back negative, and

that the treatment for both d_3 and d_4 was already suggested by other means, then the following rule can prevent using t_3 to test further:

$$Rx\_d_3 \wedge$$
$$Rx\_d_4 \ \blacktriangleright \neg g_3$$

In general, goal-setting rules shall be used to set *relevant* goals, and goal-inhibition rules shall be used to suspend relevant, but non-contributing, goals.

7.3 Strategies that Mix Diagnosis and Therapy

Working in an exploratory-corrective domain, I was particularly interested in strategies that *combine* diagnosis and repair. With few exceptions, the strategies discussed next represent "philosophies" of diagnosis-and-repair, and were identified in the trauma management domain.

7.3.1 The "Do No Harm" principle

A basic principle of medical diagnosis is avoiding the use of a diagnostic method whose consequences may be harsher than the diagnosed condition itself. For example, while a laparotomy *can* be used to diagnose renal injury, it would not be chosen for that purpose alone; rather a CT-scan or an IVP would typically be used. However, on the occasion that a laparotomy is *anyway* necessary, e.g. to treat a lacerated diaphragm, it is preferred to investigate the renal injury directly during the surgical intervention rather than spend time on a lengthy and costly CT scan procedure.

Encoding this example in GDD, three rules will be used to infer renal injury from *any* of the three alternative sources of information: CT-scan, IVP, and the surgical procedure. Choosing the means for addressing a renal injury suspicion, one must make sure that a surgical procedure is not chosen unless it is necessary anyway.

One way to do so is to let the planner figure the most efficient method given the co-present injuries and associated goals. Alternatively, goal inhibition can be used to inhibit the diagnosis of a renal injury when a need for a laparotomy is perceived.

The "do no harm" principle extends to cases where the diagnostic method is harsher than the *treatment* of the diagnosed condition. For example, while herpes encephalitis *can* be diagnosed via a brain biopsy, it is less risky to prescribe a drug for its treatment and then observe whether it works (empirical therapy)[3].

7.3.2 Buying time

If a definitive treatment requires an elaborate procedure that cannot be done immediately, there can sometimes be another procedure which although less effective, is quick and can provide a *temporary* relief. Buying time is particularly useful in patients suffering multiple injuries, where it can allow diagnosis and treatment of other injuries. Consider, for example, an unstable patient suffering a tension pneumothorax. Temporary relief can be provided by decompressing the pressure. A more definitive chest tube procedure, and if necessary an operation, can then follow. In TraumAID, such a strategy is encoded by separating two goal-setting rules: one for the urgent treatment and the other for the more definitive one. The definitive goal will only be instantiated after the urgent treatment has been accomplished.

In a slight twist, one may want to "buy time" for more accurate *diagnosis*. In a space station domain, for example, it may be easy to identify that air is leaking but hard to find the precise cause of leakage. In the meantime, air can be temporarily added to maintain habitability and allow for further investigation[4].

[3]This example was communicated to me by Chris Cimino.
[4]This example was communicated to me by Ethan Scarl.

7.3.3 All roads lead to Rome

Sometimes, it may not be known which of two or more diseases is present. However, if both share the same treatment, then further differentiation will have no effect on the choice of treatment and by the GDD principle should thus be avoided. In case of abdominal injury, for example, it may sometimes be hard to determine which of the internal organs were injured. However, since they all require a laparotomy, the diagnostic process is ceased whenever *one* organ is determined to be injured. Furthermore, a laparotomy will also be called for if there is enough *non-specific* evidence for intra abdominal injury. In GDD, that strategy is encoded via goal-inhibition rules.

7.3.4 Cover all bases

Related to the "All roads lead to Rome", is the strategy of treating *all* remaining hypotheses. Once left with a small set of such hypotheses, it may well pay to treat *as if* they were all present, rather than to refine the diagnosis. This strategy was used by MYCIN, where a set of antibiotics was chosen to cover all diagnoses above a certain threshold.

7.3.5 Gambling

It may not always be *possible* to carry out a diagnostic process to its end, or it may be too risky to do so. A decision has to be taken considering a number of competing hypotheses that have not yet been confirmed, nor eliminated. Utility theory tells us that for such a decision to be *optimal*, it should maximize the *expected utility* as measured by preferences over possible outcomes. Indeed, utilities have been used extensively in medical decision making (see [80, 116]). While some sort of utility considerations can be identified in each of the above strategies, as well as their instantiations in TraumAID 2.0, utilities are not explicitly represented in

TraumAID[5]. Following are a few more strategies, identified in TraumAID, in which probabilistic and utility-based considerations are involved:

- **Gamble on best scenario** — Consider a patient presenting unconscious from shock with a suspected pericardial tamponade and/or lacerated abdominal aorta. In the presence of some adverse condition (e.g. distended abdomen), it may not be practical to further differentiate the two hypotheses: definitive action is required.

  In practice, *regardless* of which condition is more believable, the correct strategy is to treat the patient as if he had a pericardial tamponade. The justification for that choice is given in the outcome table shown in Figure 7.4: pericardial tamponade is the only curable injury.

		Patient Has	
		Pericardial Tamponade	Lacerated Aorta
Administered Treatment	Pericardial Tamponade	lives	dies
	Lacerated Aorta	dies	dies

Figure 7.4: Outcome table

  In GDD, this strategy can be encoded by setting the goal of treating a pericardial tamponade and letting that goal override the former diagnostic goals. Note however, that we do *not* conclude the presence of a pericardial tamponade; we do not have enough evidence to support that conclusion. Rather, we conclude to treat as if it were concluded.

- **Picking order of attention** – Consider an unstable patient (a patient in shock), suffering a stab wound to the lower chest. The cause of shock may

[5] As aside, if it were to be done, a utility-theoretic modeling of decision-making in trauma management will necessarily be multi-attribute [84]. This is the result of the possible interaction of concurrent decisions for multiple injuries.

have to be urgently removed, yet *three* potential causes may be hypothesized: tension pneumothorax (due to collapse of the lungs), pericardial tamponade (due to bleeding in the pericardial sac), or abdominal shock (due to internal bleeding in the abdomen).

With insufficient information, the urgency of the situation dictates guessing the cause of shock and administering partial treatment *as if* it was the actual cause. Such guesses will be taken in a sequence until the shock is relieved. Of course, the order in which the various hypotheses are tried is important. By the utility-theoretic model, hypotheses should be tried in an order that maximizes the expected utility. In practice, the order in which potential causes are tried depends on accompanying signs, and on the expected outcome of the decision, roughly following an informal utility-theoretic decision process.

It is interesting to note that goals instantiated in this process are typically diagnostic. Yet, due to the urgency of the situation, the procedures that will be recommended will typically be therapeutic in nature but will have the diagnostic side-effect of confirming or rejecting the hypothesis, depending on their outcome. For example, a needle aspiration of the chest will be ordered for the diagnosis of a tension pneumothorax, thereby also providing an indication as to whether (a) the condition was present, and (b) was the sole source of shock. Thus, this procedure mixes diagnosis and therapy in a *reverse* order: a condition is diagnosed after, and as a result of its treatment.

Gambling strategies are considerably affected by the utilities attached to various outcomes. Naturally, such utilities are patient-specific, and thus, the appropriate strategy may differ from one patient to another. The treatment of spinal cord trauma with steroids, for example, may improve the chances of the patient walking again, but may also cause complications, and even death, in certain patients. In trauma management, using patient-specific utilities is complicated by the fact that patients can often *not* be asked for their preference.

7.3.6 Treating causal chains

Chain effects are common in physical systems. For example, a gunshot wound to the back may injure the aorta. Then, blood accumulating in the chest cavity (hemothorax) may drive the patient into an unstable condition (shock).

On such occasions, one has to decide which to address first: the problem (e.g. the hemothorax), or its cause(s) (e.g. the injured aorta). More generally, given a causal chain, one has to decide which problems on the chain to address first. Unfortunately, there is no single recipe. Such a decision is affected by a number of factors, e.g.

1. The duration and persistence of the problem and cause. In particular, whether the cause was a single event in the past, or is still present and continuing to cause the problem. Similarly, the need to address the problem depends on whether it will continue to be present, and its severity, once its cause is removed.

2. The window of opportunity for fixing the problem. If there is urgency in addressing the problem, then it may be preferable to address it first, even if it is impossible to address it completely before its cause is permanently fixed, and even if significantly more work will be necessitated that way.

3. Lack of knowledge about the cause;

4. Amenability of the cause to treatment, the risk associated with such treatment, or potential drastic side-effects of such treatment.

Given such considerations, following (and hybrids) are some possible strategies:

1. address the cause only, letting the problem take care of itself as a result of such treatment;

2. address the cause first; then take care of the problem;

3. iterate through staged therapy for both cause and problem;

4. address only the problem, ignoring or probing the cause;

5. address the problem; then the cause (possibly while maintaining treatment for the problem);

6. relieve the problem; then address the cause; then provide definitive treatment to the problem.

In GDD, we can use goal-inhibition rules to express order preference on therapeutic goals. Each of the above strategies can be encoded in GDD using such rules.

A strategy for treating causal chains is proposed by [98]. To devise treatment, explicitly represented causal chains are traversed until a treatable cause is encountered, which is then treated.

7.3.7 Causing new problems

Treatment, e.g. a drug, for a given problem may itself cause another problem, which should then be addressed. In other circumstances, the other problem is created to facilitate the treatment to the former problem, e.g. lowering the blood potassium level to stop the heart during an open heart operation. Even more common in practice is *risking* new problems, as a result of treatment prescribed to other problems.

In GDD, the prescription of problem-causing treatment should trigger (via goal-setting rules) the therapeutic goals associated with the new problem. Where there is risk of such new problems, one should make sure to instantiate diagnostic goals, aimed at appropriate monitoring.

Chapter 8

Conclusion

8.1 Summary of Contributions

This dissertation describes contributions of three types: (1) a design methodology for exploratory-corrective agents which integrates diagnostic reasoning and planning, including specific frameworks to implement each of these reasoning sub-tasks; (2) a consultation system for the multiple trauma management domain in which this methodology is implemented; and (3) a collection of abstract diagnosis-and-repair strategies which will have to be accounted for by any alternative methodology.

8.1.1 Exploratory-Corrective Management Architecture

To address the common interplay between exploratory and corrective reasoning and activity, I have proposed a reasoning architecture which integrates diagnostic reasoning and planning, and which interacts in a feedback loop with an action/perception component. In this *Exploratory-Corrective Management* (ECM) architecture, reasoning cycles between:

1. A diagnostic reasoner which is iteratively charged with characterizing what is true (the diagnosis); what needs to be determined (the diagnostic goals), and

what needs to be achieved (the corrective, or therapeutic, goals); and

2. A planner which is iteratively charged with mediating between competing diagnostic and therapeutic needs.

This architecture satisfies the following requirements of the trauma management domain:

1. It allows interleaving diagnosis and repair;

2. In the beginning of each loop, it allows the diagnostic reasoner to set diagnostic and therapeutic goals; in the end of each loop, it allows it to monitor actions and other events, verify *actual* goal achievement, and adapt goals as necessary;

3. It positions the planner to mediate between concurrent diagnostic and therapeutic needs, and also to dynamically adapt plans according to changes in goals and/or other knowledge.

8.1.2 Goal-Directed Reasoning and Diagnosis

I have presented a logical calculus for *goal-directed* reasoning in which, in addition to the standard belief measure, propositions are also assigned an *attitude* measure. An agent's attitude towards a proposition indicates the perceived relevance in acquiring information about this proposition, or of achieving a state in which this proposition holds, depending on the semantic interpretation of the particular proposition. The combination of attitude and belief for a given proposition is used to determine its goalhood.

Goal-Directed Diagnosis (GDD) is a formalization of the diagnostic task that is based on this calculus. GDD begins from the principle that *diagnosis is only worthwhile to the extent that it has the potential to affect subsequent repair decisions*, and similarly that *repair is only worthwhile to the extent that it can positively affect the*

eventual outcome. The key in the GDD formalization is that explicit representation and reasoning about goals can facilitate *focusing* on worthy exploratory-corrective activity. Situated in the ECM architecture, goals also serve as a natural interface with the planner. Furthermore, I have shown that the number of goals passed on to the planner (and thus the complexity of planning) can often be reduced via goal-level resolution within the GDD reasoner.

8.1.3 Progressive Horizon Planning

Progressive Horizon Planning (PHP) is an incremental *partial-global* planning framework in which the eventual plan is being shaped while it is executed. In particular, intermediate plans, constructed in each cycle of the ECM architecture, are expected to be partially followed before being adapted to reflect new information and goals. The intermediate plans themselves are partial in that not all goals are known, and also in that they are each constructed via an approximate plan *sketching* algorithm and then optimized to a *horizon*. Intermediate plans are, however, global in that they each address all known goals. I have presented two variations of PHP planning: one that is based on Korf's Real-Time A* planning algorithm [89], and one which is based on the TraumAID 2.0's planner. Assuming plans can be "sketched" quickly, both variations are shown to run in polynomial time.

8.1.4 Planning as Means-Selection and Ordering

I have presented a functional decomposition of the planning task into two sub-tasks:

1. *Selection* of a set of procedures/actions that parsimoniously address the current combination of goals; and

2. *Ordering* these procedures into a single overall plan, taking into account a set of preferences and constraints.

I have argued that this approach for planning is particularly useful for domains with limited interaction between goals. I have formalized the selection sub-task as a *set-covering* problem whereas the ordering sub-task has been formalized as constraint-based *scheduling*. Given the inherent dependency between the choice and respective ordering of procedures, I have presented an algorithm which interleaves the two. Within TraumAID 2.0's PHP framework, this *selection-and-ordering* algorithm serves for sketching intermediate plans.

8.1.5 A Representation Methodology for ECM Agents

I propose a methodology for effectively representing desired behavior in ECM agents. Specifically, in accordance with the way in which reasoning is decomposed in the ECM architecture, I propose

1. that *local* patterns of behavior (or strategies) be *explicitly* encoded in GDD rules, in pre-defined local procedures, and in mappings from goals to the corresponding procedures and actions;

2. that alternative choices of procedures for goals in each of these local strategies be specified and ranked by their respective preferences;

3. that constraints and preferences on *combinations* of goals, procedures, and actions, be specified; and finally

4. that given a combination of goals, these principles be used by the planner to *implicitly* merge on-line the current combination of local strategies.

8.1.6 The TraumAID 2.0 System

A GDD reasoner and a PHP planner have been implemented in TraumAID 2.0, a consultation system for the multiple trauma management domain. During its

development, TraumAID 2.0 has been tested on 270 cases that were hand-crafted to cover important features of the domain. TraumAID 2.0's plans have also been retrospectively compared to those generated by its predecessor, TraumAID 1.0, and to the actual care on 97 *real* trauma cases. Three independent trauma experts have judged TraumAID 2.0's plans preferable to TraumAID 1.0's plans by a ratio of 62:9 with 26 ties ($p < 10^{-10}$ by binomial test), and to the actual care by a ratio of 64:17 with 16 ties ($p < 10^{-7}$). In at least two cases, it is believed that a patient's death could have been prevented had TraumAID 2.0's management been followed.

8.1.7 A Collection of Diagnosis-and-Repair Strategies

Purely diagnostic strategies are much better studied and documented than strategies that mix diagnosis and repair. I have compiled a collection of *abstract* diagnosis-and-repair strategies which represent various approaches to tackling such problems. Most of these strategies were identified in the trauma domain, but appear in other domains as well. Although murky and incomplete, I found this collection intriguing, and I believe it should be of interest to other workers in medical decision making, and Artificial Intelligence in general. In particular, alternative methodologies for representation and reasoning in exploratory-corrective domains will have to be capable of representing these strategies.

8.2 Extensions and Directions for Future Research

8.2.1 Extensions to GDD

Formalizing GDD in MVL opens the door to extending the paradigm via the use of more expressive bilattices. In Section 3.8, for example, I presented a prioritized default bilattice. One way to think of the truthfulness measures associated with points in this bilattice is as qualitative *probabilities*. The calculus is somewhat odd

in that the truth value with the highest knowledge content is simply adopted while information with a lesser knowledge content is discarded, but it is nevertheless sensible. Similarly, points in a prioritized default version of the attitude bilattice can be thought of as *utility* values, indicating the perceived utility in acquiring information about the given proposition, or achieving a state in which it holds.

Extended bilattices can support more detailed models. In particular, a finer partition of the belief and attitude bilattice can support a finer reasoning by the planner about the relative relevance and current degrees of satisfaction of competing goals. As noted in Section 3.8, fairly detailed default bilattices can be constructed. Ginsberg [58] discusses possible-worlds-based bilattices. Unfortunately, if done by hand, the model-building process which is cumbersome to begin with, becomes even more so with complex bilattices. Such extensions, however, *can* maybe be useful if models can be built automatically (see Section 8.2.4). Probabilities can then be estimated from frequencies, and utilities can be assigned via a crediting/reinforcement learning process.

8.2.2 Extensions to PHP

The most obvious question in the PHP framework is how to determine the optimal horizon. While TraumAID 2.0 currently uses a 1-action horizon, no justification for this choice was provided, and there was no theoretical difficulty with extending it to an arbitrary depth. On one hand, there is an obvious computational cost associated with deeper horizons. On the other hand, a deeper horizon is likely to result in plans that are more likely to be closer to optimal[1]. Thus, it may be useful to be able to determine the appropriate *tradeoff* for a given problem.

Also, as so far described, the horizon was always set as a fixed number of actions. In reality, however, different actions may carry different weight in affecting the course of actions. Some actions, for example, are easily reversible whereas others may have

[1]Note the double use of probability in this statement.

a permanent effect on the patient. Thus, the horizon can be more generally defined in terms of a domain-specific *evaluation function*, prioritizing the space of possible plan initiations.

8.2.3 Extensions to Selection-and-Ordering Planning

In developing the Selection-and-Ordering algorithm, it was assumed that goals interact mostly via the choice and ordering of actions, that procedures and actions interact with each other via precedence and compatibility relations, and that procedures also interact with goals via time constraints. In contrast, most of the work in planning deals with much more intricate relations. An obvious question is thus whether the selection-and-ordering framework can be *gradually* extended to accommodate more complex scenarios, even if at a cost.

While for the most part, I leave this as an open question, I wish to point to an extension of the framework that can deal with sub-goals. Sub-goals are goals that when satisfied, at a certain order, result in the achievement of some high-level goal. They are useful in problems that can be broken down into sub-problems which can be solved separately.

One way to use sub-goals in the ECM architecture is to use GDD goal-setting rules to set the sub-goals from the relevance of the high-level goal, and then use a GDD evidential rule to conclude that the higher level goal is satisfied from the fact that each of the sub-goals is satisfied. Alternatively, TraumAID 2.0 allows sub-goals to be specified as part of a procedure. When the selection-and-ordering algorithm places a chosen procedure into the plan, it simply recurs on its sub-goals. Surgical procedures, for example, can often be sub-divided into pre-operative preparations, the incision, and the particular repair. The purpose of the incision is to expose the organs that need be repaired. To facilitate the choice of an incision that can serve as many purposes as possible, TraumAID 2.0, represents the incision part as a sub-goal. Figure 8.1 presents a procedure for repairing esophageal injury: first,

antibiotics have to be administered, then action has to be taken to satisfy the goal of gaining access to the right chest cavity, and then the actual repair must be done.

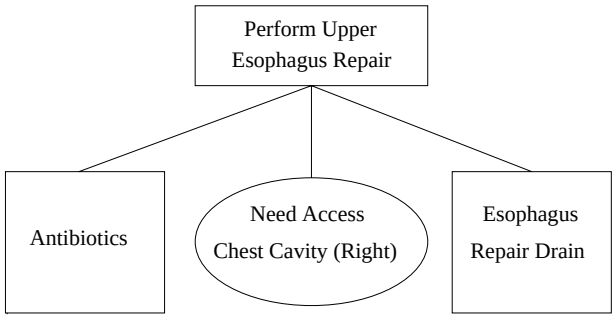


Figure 8.1: Procedure-Action Mapping for Esophagus Repair

8.2.4 Automating Knowledge Acquisition and Validation

My most important conclusion from my work on TraumAID 2.0 is that it is extremely hard to hand-construct a model of expertise. Furthermore, once constructed, it is even harder to maintain and extend such a model. This conclusion has led me to explore ways to *automate* this process. I am currently developing a protocol for recording trauma cases, or annotating existing ones, which will allow applying standard machine learning techniques to automatically infer GDD-style models and test their validity.

In short, the protocol records observations, conclusions, goals, and actions, in the order in which they were reported, made, and taken respectively. Figure 8.2 presents an example of part of such protocol. These case descriptions are first broken into a number of *scenarios*, each corresponding to a temporal snapshot. Each of the scenarios describes an initial attitude-belief, the conclusions made, goals set, and the next action to be taken. Every new action potentially defines a new scenario. The purpose of learning is to generalize a mapping from a scenario's initial attitude-belief to the set of conclusions (evidential rules) and goals (goal-setting rules). While this protocol also records the actions taken, I have not yet tackled the problem of

automatically acquiring planning knowledge[2].

8.2.5 Extensions of TraumAID

Other Special Forms of Reasoning

Part of the motivation for this work was the fact that rules proved to provide poor means for representing and reasoning about actions. The solution proposed in this work was to modularize reasoning into that which is diagnostic in nature and that in which a course of action has to be chosen. More research is needed to identify areas of reasoning that can benefit from specialization, and to devise and implement solutions which can be *integrated* into the current system. Current areas of research in the TraumAID project include

1. Quantitative and qualitative models of the cardiovascular system which given the state of the patient can *project* blood loss and blood pressure figures in support of pre-operative decisions [110].

2. Spatial reasoning, using a 3D model of the body and kinematic models of bullet and knife penetration to determine which internal organs have potentially been injured in a more principled (and thus more scalable) fashion [83].

Improving the Delivery of Advice

For TraumAID's advice to be followed, it must win the physician's cooperation. Recall (Chapter 4.1) that gaining the physician's *confidence* was an important factor in preferring complete plans. Significant deficiencies in TraumAID 1.0's interface which were observed when it was first fielded in the emergency room are currently being

[2] While given a collection of scenarios it is *theoretically* simple to acquire a next-action model, it seems much harder to acquire the knowledge necessary to construct complete plans.

improved upon in a new HyperCard-based interface; speech-based input devices are also being experimented with.

More related to my work on planning is current work in the area of critiquing [56]. In many cases, residents commented that TraumAID's advice only *reassured* their opinion. Thus, rather than telling the physician what s/he already know, a different mode of interaction can be adopted in which the physician is only told when his plan deviates *significantly* from TraumAID's.

8.2.6 Other Exploratory-Corrective Domains

While developed specifically for the multiple trauma management domain, I believe the knowledge representation and reasoning methodology developed throughout this dissertation can be applied in other exploratory-corrective domains as well. In addition to being exploratory-corrective, domains for which the methodology will fit best are likely to share more features with the multiple trauma management domain. I will not repeat these features here; they are discussed when JR's case is first presented, and then elaborated upon in the appropriate technical chapters. I wish, however, to mention two important assumptions which *limit* the use of the proposed framework:

- First, I assume that the domain allows explicit formalization of the relationship between beliefs, or what is known, and goals, both diagnostic or corrective. Specifically, it is required that given a set of observations and inferred knowledge, a corresponding agenda defining all relevant information and physical goals is well defined.

- Second, I assume a limited interaction between multiple goals. I further assume that a plan for satisfying each of these goals (or multiple alternative such plans) is readily available, or can at least be computed without much effort.

> Put differently, I assume that planning is really a matter of choosing and coordinating activities aimed at various goals.

Consider, for example, an autonomous robot who is charged with the diagnosis and repair of problems aboard a submarine, or maybe a space station. Being also charged with diagnosis, our robot will typically not be told what is the problem that has to be addressed, but rather what symptoms it may have caused, e.g. "there is a reduction in the level of oxygen in room B2", or "there is loss of power in the right engine". Our robot will have to first find the source of the problem, and then act to repair it. For this to happen, a local strategy will have to be written, using goal-setting and evidential rules, to guide the robot through the investigation of the problem until it is pinpointed, and then until it is repaired.

Our robot, however, may conceivably face multiple such problems at once; in wartime, for example, our submarine may be hit by several enemy's rockets. Alternatively, a strategy for a single problem may consist of multiple parallel paths represented in multiple concurrent goals, e.g. investigating multiple alternative diagnoses, or partial repair and continued refinement of a diagnosis. While local plans, or procedures, for addressing each of the goals is assumed to be explicitly specified, it is virtually impossible to pre-specify a composite plan for every possible combination of goals. Thus, a set of high level principles must be used in combining the appropriate procedures. Precedence constraints implied by notions of urgency and priority, or the logistic preference of avoiding having the agent run back and forth between rooms, can be implemented as in TraumAID 2.0. Other constraints, such as on the cargo the agent may carry, can be implemented using compatibility constraints.

```
NEW_CASE
GIVEN
     Radiography Available
     Wound[Gunshot, Sternal, Unknown]
GIVEN_NEG
     Shock
     Distended Neck Veins
     Muffled Heart Sounds
     Hemoptysis
CONCLUDE
     Chest Wall Penetration
PROJECTED_GOALS
     Potential Occult Injury Pleural Space[Right]
     Potential Occult Injury Pleural Space[Left]
     Clinical Possibility Of Bronchial Injury[Right]
     Clinical Possibility Of Bronchial Injury[Left]
     Need Survey Chest X Ray Airway
     Rx Chest Wall Penetration
GIVEN
     Cover Wound Occlusive Dressing
GOALS_PURSUED
     Rx Chest Wall Penetration
GIVEN
     X Ray Bullet In Chest Midline
     Bullet Image[Chest Midline, Unknown LAT, 1]
     Survey Chest X Ray (other results normal)
CONCLUDE
     Thru Wounds[0]
     Potential Occult Injury Pleural Space[Left]
     Potential Occult Injury Pleural Space[Right]
PROJECTED_GOALS
     RO Bullet In Mediastinum
     Need Lateral Chest X Ray To Localize Bullet
     Rx Observable Potential Pleural Injury[Left]
     Rx Observable Potential Pleural Injury[Right]
GOALS_PURSUED
     Need Survey Chest X Ray Airway
```

Figure 8.2: Example of Protocol for Reporting a Trauma Case

Bibliography

[1] United States Department of Health and Human Services. *The International Classification of Diseases*, 9th revision, Clinical Modification (ICD 9 CM), 2nd edition. United States Government Printing Office, Washington DC, 1980.

[2] Subcommittee on Advanced Trauma Life Support of the American College of Surgeons Committee on Trauma: *Advanced Trauma Life Support for Physicians. Instructors Manual.* Chicago, American College of Surgeons, 1984.

[3] National Research Council and Institute of Medicine. *Injury in America.* National Academy Press, Washington DC, 1985.

[4] Allemang, D., Tanner, M. C., Bylander, T. and Josephson, J., Computational Complexity of Hypothesis Assembly. *Proc. IJCAI-89*, Detroit, MI, 1989, pp.1112-1117.

[5] Alpay, L. L., *Modeling Medical Diagnostic Processes* Ph. D. Thesis. Centre for Information Technology in Education, The Open university, England, 1991.

[6] Agre, P. and Chapman, D., Pengi: An Implementation of a Theory of Activity. *Proc. AAAI-87*, Seattle WA, August 1987, pp. 268-272.

[7] Barret, A., and Weld, D. S., Characterizing Subgoal Interactions for Planning. *Proc. IJCAI-93*, pp. 1388-1393, Chambery, France, 1993.

[8] Ben Bassat, M., Carlson, R. W., Puri, V., Davenport, J., Schriver, J, Mohammed, L., Smith, R., Portigal, L., Lipnik, E., and Weil, M., Pattern-Based Interactive Diagnosis of Multiple Disorders: the MEDAS system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2, pp. 148-160, 1980.

[9] Boyd, C. R., Tolson, M. A., and Copes, W. S., Evaluating Trauma Care: The TRISS Method. *Journal of Trauma*, 27(4), pp. 370-378, 1987.

[10] Brooks, R. A., Intelligence without Representation. *Artificial Intelligence* 47, pp. 139-159, 1991.

[11] Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J., The Computational Complexity of Abduction. *Artificial Intelligence* 49, (1-3), pp. 25-60, 1991.

[12] Champion, H. R., Sacco, W. J., Carnazzo, A. J., *et al.* Trauma Score, *Critical Care Medicine*, 9, pp. 672-676, 1981.

[13] Champion, H. R., Copes, W. S., Sacco, W. J., Lawnick, M. M., Bain, L. W., Gann, D. S., Gennarelli, T., Mackenzie E., and Schwaitzberg, S., A New Characterization of Injury Severity. *Journal of Trauma*, 30(5), pp. 539-546, 1990.

[14] Chapman, D., *Planning for Conjunctive Goals.* Masters Thesis, MIT Laboratory for Artificial Intelligence, Cambridge, MA, 1985.

[15] Cheng, J., and Irani, K. B., Ordering Problem Subgoals. *Proc. IJCAI-89*, Detroit, MI, pp. 931-936, 1989.

[16] Chvatal, V., A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3), 1979, pp. 233-235.

[17] Clarke, J. R., Sachdeva, A. K., Nieman, L., and Gracely, E. J., What Chief Surgical Residents May Still Need to Learn. *Unpublished abstract.*

[18] Clarke, J. R., Cebula, D., and Webber, B. L., A Computerized Decision Aid for Trauma, *Journal of Trauma*, 28, 1988, pp. 1250-1254.

[19] Clarke, J. R., Niv, M., and Webber, B. L., Computerized Patient-Specific Protocols for Managing Penetrating Chest Injuries (abstract). *Theoretical Surgery*, 5(3), 1990, pp. 148-149.

[20] Clarke, J. R., Rymon, R., Webber, B. L., Hayward, C., Santora, T., Wagner, D., Friedman, C., Ruffin, A., Blank, C., Nieman, L., and Eastman, A. B., Computer-Generated Protocols: A Tool for Quality Control During the Initial Definitive Management of Trauma Patients. Initial Trauma Management Plans. Submitted to the *Annual Meeting of the American Association for the Surgery of Trauma*, September 1993.

[21] Clancey, W. J., The Epistemology of a Rule-Based Expert System – A Framework for Explanation. *Artificial Intelligence*, 20, 1983, pp. 215-251.

[22] Clancey, W. J., and Shortliffe, E. H., *Readings in Artificial Intelligence: The First Decade* Addison-Wesley, Reading MA, 1984.

[23] Clancey, W. J., Acquiring, Representing, and Evaluating a Competence Model of Diagnostic Strategy. *STAN-CS-85-1067*, Department of Computer Science, Stanford University, 1985.

[24] Clancey, W. J., Heuristic Classification. *Artificial Intelligence*, 27, 1985, pp. 289-350.

[25] Clendening L., and Hashinger, E.H., *Methods of Diagnosis*, Mosby, St. Louis, MO, 1947.

[26] Console L., and Torasso, P., Hypothetical Reasoning in Causal Models. *International Journal of Intelligent Systems*, 5, pp. 83-124, 1990.

[27] Console L., and Torasso, P., An Approach to Diagnosis on Causal-Temporal Models. *Proc. AAAI Spring Symposium*, AI in Medicine, Stanford, CA, 1990, pp. 37-41.

[28] Console, L., and Torasso, P., A Spectrum of Definitions of Model-Based Diagnosis. *Computational Intelligence*, 7, pp. 133-141, 1991.

[29] Davis, R., Diagnostic Reasoning Based on Structure and Behavior. *Artificial Intelligence*, 24, 1984, pp. 347-410.

[30] Davis, R., Buchanan, B. G., and Shortliffe, E. H., Retrospective on "Production Rules as a Representation for a Knowledge-based Consultation Program". *Artificial Intelligence*, 59, 1993, pp. 181-189.

[31] Dean, T., and Wellman, M. P., *Planning and Control.* Morgan-Kaufman, 1991.

[32] deKleer, J., Doyle, J., Steele, G. L., and Sussman, G. J., Explicit Control of Reasoning. *MIT AI Memo No. 427*, 1977 (also appears in *Artificial Intelligence: An MIT Perspective*, Vol 1, ed. Winston, pp. 93-118, MIT Press, 1979.

[33] deKleer, J., Doyle, J., Rich, C., Steele, G. L., and Sussman, G. J., AMORD: A Deductive Procedure System. *MIT AI Memo No. 435*, 1978.

[34] de Kleer, J. and Williams, B., C., Diagnosing Multiple Faults. *Artificial Intelligence*, 32, 1987, pp. 97-130.

[35] de Kleer, J. and Williams, B., C., Diagnosis with Behavioral Modes. *Proc. IJCAI-89*, Detroit, MI, pp. 1324-1330, 1989.

[36] de Kleer, J., Mackworth, A. K., and Reiter, R., Characterizing Diagnoses. *Proc. AAAI-90*, pp. 324-330, Boston, MA, 1990.

[37] de Kleer, J., Focusing on Probable Diagnoses. *Proc. AAAI-91*, pp. 842-848, Anaheim CA, 1991.

[38] de Kleer, J., An Improved Incremental Algorithm for Generating Prime Implicates. *Proc. AAAI-92*, pp. 780-785, San Jose CA, 1992.

[39] Dowie, J., and Elstein, A., *Professional Judgement – A Reader in Clinical Decision Making.* Cambridge University Press, 1988.

[40] Drummond, M., and Curie, K., Goal Ordering in Partially Ordered Plans. *Proc. IJCAI-89*, pp. 960-965, Detroit, MI, 1989.

[41] Elkan, C., Incremental, Approximate Planning. *Proc. AAAI Spring Symposium on Planning in Uncertain Unpredictable and Changing Environments*, Stanford CA, 1990.

[42] Elstein, A. S., Shulman, L. S., and Sprafka, S. A., *Medical Problem Solving: An Analysis of Clinical Reasoning.* Harvard University Press, Cambridge, MA, 1978.

[43] Ernst, G., and Newell, A., GPS: A Case Study in Generality and Problem Solving. *Academic Press*, New York, 1969.

[44] Fikes, R. E., and Nilsson, N. J., STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2, 1971, pp. 189-208.

[45] Fikes, R. E., Hart, P. E., and Nilsson, N. J., Learning and Executing Generalized Robot Plans. *Artificial Intelligence* 3, 1972, pp. 251-288.

[46] Fink, E., and Yang, Q., Automatically Abstracting the Effects of Operators. *Proc. 1st Int'l Conf. on AI Planning Systems*, College Park MD, pp. 243-251, 1992.

[47] Firby, J., An Investigation into Reactive Planning in Complex Domains. *Proc. AAAI-87*, Seattle WA, pp. 677-682, 1987.

[48] Foulser, D. E., Li, M., and Yang. Q., A Qualitative Theory for Plan Merging. *Proc. AAAI-91*, Anaheim CA, 1991, pp. 673-678.

[49] Fox, M. S., and Smith, S. F., ISIS: A Knowledge-based System for Factory Scheduling. *Expert Systems*, 1(1), pp. 25-49, 1984.

[50] Friedrich, G., Gottlob, G., Nejdl, W., Physical Impossibility Instead of Fault Models. *Proc. AAAI-90*, Boston MA, 1990, pp. 331-336.

[51] Friedrich, G., Gottlob, G., and Nejdl W., Formalizing the Repair Process. *2nd International Workshop on Principles of Diagnosis*. Milan, Italy, Oct 1991, pp. 11-22.

[52] Friedrich, G., and Nejdl W., Choosing Observations and Actions in Model-Based Diagnosis/Repair Systems. *Proc. 3rd International Workshop on Principles of Diagnosis*. Seattle WA, 1992, pp. 76-85. Also appears in *Proc. Third International Conference on Principles of Knowledge Representation and Reasoning*, Boston MA, 1992, pp. 489-498.

[53] Garey, M. R. and Johnson, D. S., *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979, pp. 46, 54-56.

[54] Genesereth, M. R., The Use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence*, 24, 1984, pp. 411-436.

[55] Georgeff, M. P. and Lansky, A., Reactive Reasoning and Planning. *Proc. AAAI-87*, Seattle WA, 1987, pp. 677-682.

[56] Gertner, A., Real-Time Critiquing of Integrated Diagnosis/Therapy Plans. *Proc. AAAI Workshop on Expert Critiquing Systems*, Washington DC, 1993.

[57] Ginsberg, M. L., Counterfactuals. *Artificial Intelligence*, 30, 1986, pp. 35-79.

[58] Ginsberg, M. L., Multivalued Logics: A Uniform Approach to Inference in Artificial Intelligence. *Computational Intelligence*, 4:265-316, 1988.

[59] Ginsberg, M. L., and Smith, D. E., Reasoning about Action I: A Possible Worlds Approach. *Artificial Intelligence*, 35, 1988, pp. 165-195.

[60] Glass, R.I., New Prospects for Epidemiologic Investigations. *Science (234)*, pp. 951-959, 1986.

[61] Ginsberg, M. L., Universal Planning: An (Almost) Universally Bad Idea. AI Magazine, Vol. 10, Number 4, Winter 1989, pp. 40-44.

[62] Ginsberg, M. L., Defaults and Hierarchical Problem Solving. Technical Report, Computer Science Department, Stanford University, 1990.

[63] Green, C., Application of Theorem Proving to Problem Solving. *Proc. IJCAI-69*, pp. 741-747, Washington DC, 1969. Also in *Readings in Planning*, Allen *et al.* eds., pp. 67-87, 1990.

[64] Greiner, R., Smith, B. A., and Wilkerson R. W., A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41, pp. 79-88, 1989.

[65] Hammond, K. J., *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, New York, 1989.

[66] Hart, P. E., Nilsson, N. J., and Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, 4(2), pp. 100-107, 1968.

[67] Hayes, C., Using goal-interactions to Guide Planning. *Proc. AAAI-87*, Seattle WA, 1987, pp. 224-228.

[68] Hayes-Roth, B., Guardian. A Prototype intelligent Agent for Intensive-Care Monitoring. *Artificial Intelligence in Medicine*, 4(2), pp. 165-185, 1992.

[69] Herskovits, E., *Computer-Based Probabilistic-Network Construction.* Ph. D. Thesis, Departments of Computer Science and Medicine, Stanford University, CA, 1991.

[70] Hickam, D. H., Shortliffe, E. H., Bischoff, M. B., Scott, A. C., and Jacobs, C. D., The Treatment Advice of a Computer-Based Cancer Chemotherapy Protocol Advisor. *Annals of Internal Medicine*, 103, pp. 928-936, 1985.

[71] Hilliard, J., Foreword to *Medical Problem Solving: An Analysis of Clinical Reasoning*. Elstein et al. eds., Harvard University Press, Cambridge, MA, 1978.

[72] Holzblatt, L. J., Diagnosing Multiple Failures Using Knowledge of Component States. *Proc. IEEE AI Applications*, pp. 139-143, 1988.

[73] Horvitz, E. J., Heckerman, D. E., Nathwani, B. N. and Fagan, L. M., Diagnostic Strategies in the Hypothesis-Directed PATHFINDER System. *First Conference on AI Applications*, 1984, pp. 630-636.

[74] Hsu, J. Y., Partial Planning with Incomplete Information. *AAAI Spring Symposium on Planning in Uncertain, Unpredicatable and Changing Environments*, Stanford CA, 1990, pp. 62-66.

[75] Johnson, D. S., Approximation Algorithms for Combinatorial Problems. *Journal of Computer System Science*, 9, pp. 256-278, 1974.

[76] Kaelbling, L. P., An Architecture for Intelligent Reactive Systems. In *Reasoning about Actions and Plans*, Lansky and Georgeff eds., pp. 395-410, Morgan Kaufman, 1987.

[77] Kahneman, D., Slovic, P., and Tversky, A., *Judgement under Uncertainty: Heuristics and Biases* Cambridge University Press, Cambridge UK, 1982.

[78] Karp, R. M., Reducibility Among Combinatorial Problems. In Miller and Thatcher eds., *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-103, 1972.

[79] Kassirer, J. P., and Gorry, A., Clinical Problem Solving: A Behavioral Analysis. *Annals of Internal Medicine*, 89, pp. 245-255, 1978.

[80] Kassirer, J. P., Moskowitz, A. J., Lau, J., and Pauker, S. G., Decision Analysis: A Progress Report. *Annals of Internal Medicine*, 106:275-291, 1987.

[81] Kassirer, J. P., and Kopelman, R. I., *Learning Clinical Reasoning.* Williams and Wilkins, Baltimore MD, 1991.

[82] Kautz H. A., *A Formal Theory of Plan Recognition.* Ph.D. Thesis, University of Rochester, 1987.

[83] Kaye, J., *Trauma Modeling of Anatomy and Physiology.* Thesis Proposal, University of Pennsylvania, 1993.

[84] Keeney, R. L., and Raiffa, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, New York, 1976.

[85] Knoblock, C. A., Learning Abstraction Hierarchies for Problem Solving. *Proc. AAAI-90*, Boston MA , 1990, pp. 923-928.

[86] Korf, R. E., Operator Decomposability: A New Type of Problem Structure. *Proc. AAAI-83.*

[87] Korf, R. E., Depth-First Iterative Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27, pp. 97-109, 1985.

[88] Korf, R. E., Planning as Search: A Quantitative Approach. *Artificial Intelligence*, 33, pp. 65-88, 1987.

[89] Korf, R. E., Real-time Search for Dynamic Planning. *Proc. AAAI Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments*, Stanford CA, 1990, pp. 72-76.

[90] Korf, R. E., Real-time Heuristic Search. *Artificial Intelligence*, 42, pp. 189-211, 1990.

[91] Korf, R. E., Linear-Space Best-First Search: Summary of Results. In *Proc. AAAI-92*, San Jose CA, 1992.

[92] Langlotz, C. P., Fagan, L. M., Tu, S. W., Sikic, B. I., and Shortliffe E. H., Combining Artificial Intelligence and Decision Analysis for Automated Therapy Planning Assistance. *Proc. of MEDINFO 86*, pp. 794-798, 1986.

[93] Lansky, A. L., Localized Event-Based Reasoning for Multi-Agent Domains. *Computational Intelligence*, 4, pp. 319-340, 1988.

[94] Lansky, A. L., Localized Search for Multiagent Domains. *Proc. IJCAI-91*, Sydney Australia, pp. 252-258, 1991.

[95] Leake, D., *Evaluating Explanations: A Content Theory.* (Chapter 9) Lawrence Earlbaum, Hillsdale, NJ, 1992.

[96] Ledley, R. S., and Lusted, L. B., Reasoning Foundations of Medical Diagnosis, *Science*, Vol. 130 (3366), pp. 9-21, 1959.

[97] Levesque, H., A Knowledge-Level Account of Abduction. *Proc. IJCAI-89*, Detroit MI, 1989.

[98] Long, W., Naimi, S., and Criscitiello, M. G., A Knowledge Representation for Reasoning about the Management of Heart Failure. *Proc. Computers in Cardiology Conference*, pp 373-376, 1982.

[99] Markley, J., Masters work, Department of Computer Science, University of Pennsylvania, 1991.

[100] McAllester, D., and Rosenblitt, D., Systematic NonLinear Planning. *Proc. AAAI-91*, Anaheim CA, 1991, pp. 634-639.

[101] McCarthy, J., and Hayes, P., Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4, pp. 463-502, 1969.

[102] Miller, G. A., The Magical Number Seven, plus or minus two: Some Limits on Our Capacity for Processing Information. *Psychology Review*, 63, pp. 81-97, 1956.

[103] Miller, R. A., Pople, H. E. and Myers, J. D., INTERNIST-I, An Experimental Computer Based Diagnostic Consultant for General Internal Medicine. *New England Journal of Medicine*, 307(8), 1982, pp. 468-476.

[104] Miller, P. L., ATTENDING: Critiquing a Physician's Management Plan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(5), pp. 449-461, 1983.

[105] Minton, S., Selectively Generalizing Plans for Problem Solving. *Readings in Planning*, Allen *et al.* eds., pp. 651-654, 1990.

[106] Minton, S., Drummond, M., Bresina, J., and Philips, A., Total Order vs. Partial Order Planning: Factors Influencing Performance. *Proc. KR-92*, Cambridge, MA, 1992, pp. 83-92.

[107] Moore, E. E., Eiseman, B., and van Way, C. W., *Critical Decisions in Trauma*. Mosby, St. Louis MO, 1984.

[108] Muscettola, N., and Smith, S. F., Integrating Planning and Scheduling to Solve Space Mission Scheduling Problems. *Proc. Workshop on Innovative Approaches to Planning, Scheduling, and Control*, San Diego CA, 1990, pp. 220-230.

[109] Nau, D. S., Hendler, J., and Yang, Q., Optimization of Multiple-Goal Plans with Limited Interaction. *Proc. Workshop on Innovative Approaches to Planning, Scheduling, and Control*, San Diego CA, 1990, pp. 160-165.

[110] Neumann, S., *Modeling Acute Hemorrhage in the Human Cardiovascular System*. Thesis Proposal, University of Pennsylvania, 1993.

[111] Newell, A, and Simon, H. A., GPS, A Program That Simulates Human Thought. In E. A. Feigenbaum & J. Feldman eds. *Computers and Thought*, pp.

279-293, R. Oldenbourg KG, 1963. Also in *Readings in Planning*, Allen *et al.* eds., pp. 59-66, 1990.

[112] Patil, R. S., Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis. *Ph. D. Thesis*, MIT LCS TR-267, Oct 1981.

[113] Patil, R. S., Szolovits, P., and Schwartz, W. B., Modeling Knowledge of the Patient in Acid-Base and Electrolyte Disorders. in *Artificial Intelligence in Medicine*, P. Szolovits ed., Westview Press, 1982.

[114] Pauker, S. G. and Kassirer, J. P., Therapeutic Decision Making: A Cost-Benefit Analysis. *New England Journal of Medicine*, 293(5), 1975, pp. 229-234.

[115] Pauker, S. P., and Pauker, S. G., Prenatal Diagnosis: A Directive Approach to Genetic Counseling Using Decision Analysis. *Yale Journal of Biology and Medicine*, 50, 1977, pp. 275-289.

[116] Pauker, S. G., and Kassirer, J. P., Medical Progress – Decision Analysis. *New England Journal of Medicine*, Vol. 316, No. 5, pp. 250-258, 1987.

[117] Pearl, J., Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, CA, 1988.

[118] Pepper, J., and Kahn, G. S., Repair Strategies in a Diagnostic Expert System. *Proc. IJCAI-87*, Milano, Italy, 1991, pp. 531-534.

[119] Poole, D. L., Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence*, 5, pp. 97-110, 1989.

[120] Poole, D. and Provan G. M., What is an Optimal Diagnosis? *Conference on Uncertainty in Artificial Intelligence*, pp. 46-53, 1990.

[121] Poole, D., and Provan G. M., Use and Granularity in Consistency-Based Diagnosis. *2nd International Workshop on Principles of Diagnosis.* Milan, Italy, Oct 1991, pp. 1-10.

[122] Poole, D., Representing Diagnostic Knowledge for Probabilistic Horn Abduction. *Proc. IJCAI-91*, Sydney, Australia, 1991.

[123] Pople, H. E. Jr., Heuristic Methods for Imposing Structure on Ill-Structured Problems:: The Structuring of Medical Diagnosis. in *Artificial Intelligence in Medicine*, P. Szolovits ed., Westview Press, 1982.

[124] Provan, G. M., and Poole, D., The Utility of Consistency-Based Diagnostic Techniques. *Proc. KR-91*, Cambridge, MA, 1991, pp. 461-472.

[125] Provan, G. M., and Clarke, J. R., Dynamic Network Construction and Updating Techniques for the Diagnosis of Acute Abdominal Pain. *Submitted to IEEE-PAMI*, 1992

[126] Raiffa, H., *Decision Analysis*, Addison-Wesley, Reading MA, 1968.

[127] Raiman, O., de Kleer, J., Saraswat, V., and Shirley, M., Characterizing Non-Intermittent Faults. *Proc AAAI-91*, Anaheim CA, 1991, pp. 849-854.

[128] Rasmusen, E., *Games and Information: An Introduction to Game Theory.* Basil Blackwell, Oxford UK, 1989.

[129] Reggia, J. A., Nau, D. S. and Wang, P. Y., A Formal Model of Diagnostic Inference. I. Problem Formulation and Decomposition. II. Algorithmic Solution and Application. *Information Sciences 37*, 1985, pp. 227-285.

[130] Reiter, R., A Logic for Default Reasoning. *Artificial Intelligence*, 13, 1980, pp. 81-132.

[131] Reiter, R., A Theory of Diagnosis From First Principles. *Artificial Intelligence*, 32, 1987, pp. 57-95.

[132] Rushby, J. and Crow, J., Model-Based Reconfiguration: Toward an Integration with Diagnosis. *Proc. AAAI-91*, Anaheim, CA, 1991, pp. 836-841.

[133] Russell, S. J., and Wefald, E. *Do the Right Thing : Studies in Limited Rationality* MIT Press, Cambridge MA, 1991.

[134] Russell, S. J., Efficient Memory-Bounded Search Algorithms. *Proc. ECAI-92*, Vienna, Austria, 1992.

[135] Rymon, R., Webber, B. L. and Clarke, J. R., Towards Goal-directed Diagnosis (Preliminary Report). *Proc. 2nd International Workshop on Principles of Diagnosis*. Milan, Italy, 1991, pp. 23-39.

[136] Sacerdoti, E. D., Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence*, 5, 1974, pp. 115-135.

[137] Sacerdoti, E. D., *A Structure of Plans and Behavior.* New York: American Elsevier, 1977.

[138] Schoppers, M. J., *Representation and Automatic Synthesis of Reaction Plans.* Computer Science Department, University of Illinois, 1989.

[139] Schoppers, M. J., and Linden, T., The Dimensions of Knowledge-Based Control Systems and the Significance of Metalevels. *Proc. AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, Stanford CA, 1990.

[140] Shortliffe, E. H., MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection. Ph. D. Thesis, Stanford, CA, 1974.

[141] Shwe M., Blackford, M., Heckerman, D., Henrion, M., Horvitz, E., Lehman, H. and Cooper, G., A Probabilistic Reformulation of the Quick Medical Reference System. *SCAMC-90*, Symposium on Computer Applications in Medical Care, November 1990, Washington D.C., pp. 790-794.

[142] Stefik, M., Planning with Constraints (MOLGEN: Part I). *Artificial Intelligence*, 16, 1981, pp. 111-140.

[143] Struss, P., and Dressler, O., "Physical Negation" – Intergrating Fault Models into the General Diagnostic Engine. *Proc. IJCAI-89*, Detroit MI, pp. 1318-1323.

[144] *A Computer Model of Skill Acquisition.* Artificial Intelligence Series 1, American Elsevier, New York, 1975.

[145] Sycara, K. editor, Proceedings of DARPA Workshop on Innovative Approaches to Planning Scheduling and Control, San Diego CA, November 1990.

[146] Sycara K. editor, *IEEE Trans. on Systems, Man, and Cyberbetics*, special issue on planning scheduling and control, Vol 23, Number 6, 1993.

[147] Szolovits, P., *Artificial Intelligence in Medicine*, Westview Press, 1982.

[148] Szolovits, P., and Pauker, S. G., Categorical and Probabilistic Reasoning in Medicine Revisited. *Artificial Intelligence*, 59, 1993, pp. 167-180.

[149] Sun, Y., and Weld, D., Beyond Simple Observation: Planning to Diagnose. *Proc. 3rd International Workshop on Principles of Diagnosis.* Seattle WA, 1992.

[150] Tate, A., Generating Project Networks. *Proc. IJCAI-77*, pp. 888-900, 1977.

[151] Teasdale, G., and Jennet, B., Assessment of Coma and Impaired Consciousness: A Practical Scale. *Lancet*, 2, 1974, pp. 81-84.

[152] Tenenberg, J. D., *Abstraction in Planning.* Ph. D. Thesis, Computer Science Department, University of Rochester, 1988.

[153] Trunkey, D., Trauma. *Scientific American*, 249, pp. 28-35, 1983.

[154] Vere, S. A., Planning in Time: Windows and Durations for Activities and Goals. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(3), pp. 246-267, 1983.

[155] Webber, B. L., Rymon, R. and Clarke, J. R., Flexible Support for Trauma Management through Goal-Directed Reasoning and Planning. *Artificial Intelligence in Medicine*, 4(2), pp. 145-163, 1992.

[156] Weiss, S. M., Kulikowski, C. A., and Amarel, S., A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, 11, 1978, pp. 145-172.

[157] Wellman, M. P. and Doyle, J., Modular Utility Representation for Decision Theoretic Planning *Proc. 1st Int'l Conf. on AI Planning Systems*, College Park MD, pp. 236-242, 1992.

[158] West, J.G., Trunkey, D. and Lim, R.C., Systems of Trauma Care: A Study of Two Counties. *Archives of Surgery*, 114, pp. 455-460, 1979.

[159] Wilkins, D. E., Domain Independent Planning: Representation and Plan Generation. *Artificial Intelligence*, 22, 1984, pp. 269-301.

[160] D. E. Wilkins, Can AI Planners Solve Practical Problems? *Computational Intelligence*, 6(4), pp. 232-246, 1990.

[161] Winslett, M., Reasoning about Action Using a Possible Models Approach. *Proc. AAAI-88*, pp. 89-93, St. Paul, MN, 1988.

[162] Yang, Q., Nau, D. S., and Hendler, J., Merging Separately Generated Plans with Restricted Interactions. *Computational Intelligence*, 8(4), pp. 648-676, 1992.

[163] Zilberstein, S., and Russell, S. J., Efficient Resource-Bounded Reasoning in AT-RALPH. *Proc. 1st Int'l Conf. on AI Planning Systems*, College Park MD, pp. 260-266, 1992.