ADVANCED DECISION SYSTEMS

TR 1113-1

OPERATIONS MONITORING ASSISTANT SYSTEM DESIGN

Final Report

Prepared by:

J.R. Payne

G.L. Stachnick

J. Rosenschein

D.G. Shapiro

July 1986

Government Contract No. DAAB07-85-C-K571

ADS Project No. 1113

Prepared for:

Commander, U.S. Army Communications Command

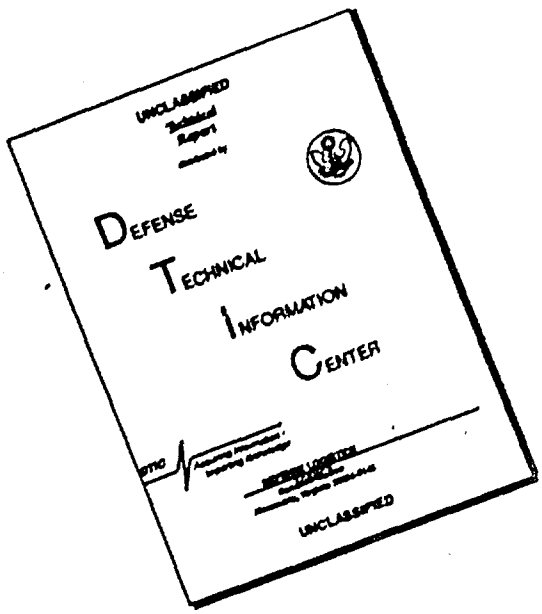
ATTN: DRSEL-PC-CM-E

SBIR Program

Ft. Monmouth, NJ 07703-5008

201 San Antonio Circle, Suite 286
Mountain View, California 94040-1289
415/941-3912 FAX: 415/949-4029

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

U. :LASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR 1113-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Advanced Decision Systems 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army CECOM AMSEL-TCS-CR
6c. ADDRESS (City, State and ZIP Code) 201 San Antonio Circle, Suite 286, Mt. View, CA 94040	7b. ADDRESS (City, State and ZIP Code) Ft. Monmouth, NJ 07703-5008
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Same as 7a 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAB07-85-C-K574
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO
11. TITLE (Include Security Classification) Operations Monitoring Assistant System Design	
12. PERSONAL AUTHOR(S) J.R. Payne, G.L. Stachnick, J. Rosenchein, D.G. Shapiro	
13a. TYPE OF REPORT ~~Draft~~ Final	13b. TIME COVERED FROM 13Sept.85 TO 1Aug.86
14. DATE OF REPORT (Yr., Mo., Day) July 1986	15. PAGE COUNT
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Operations Monitoring, Mixed-initiative, Planning, Plan execution monitoring, Artificial Intelligence,

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Corps level operations monitoring is characterized in terms of plan representations, current situation appraisals and representation, identifying variances between the two, and identifying opportunity and risk situations. A mixed-initiative operations monitoring assistant (OMA) system is designed that combines operations research, artificial intelligence, and human reasoning techniques and constructs.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ OTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE. UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

LIST OF FIGURES

STATEMENT "A" per Eli Beane
CECOM Center for Command, Control &
Communications Systems/AMSEL-RD-C3-IR,
Ft. Monmouth, NJ 07703
TELECON 5/7/90 VG

LIST OF TABLES

1. INTRODUCTION & SUMMARY

The objective of this SBIR Phase I study is to design an Operations Monitoring Assistant (OMA) system for supporting Corps G3 Operations personnel in monitoring operations and assessing the impact of events on the current operations plan.

The research effort consisted of:

- Gaining an understanding of corps level operations monitoring
  - Interaction with Army officers (Active and retired)
  - Study of Field Manuals and other pertinent documents
  - Study of scenarios
  - Observation of a Command Post Exercise (CPX)
  - Analysis of other battle management projects in ADS, BDM, and elsewhere.
  - Identifying desirable features for an OMA system.
- Reviewing applicable AI technology such as:
  - Planning
  - Representation of knowledge
  - Reasoning mechanisms (e.g., inferencing)
  - Knowledge-based system control
  - User-machine interface
- Designing an OMA system
  - Plan representations
  - Situation appraisal representations

- Reasoning processes to note:
  - Differences
  - Opportunities
  - Risk situations
- Mixed-initiative
  - Control
  - User interface

The approach did not explicitly include hardware or embedding the OMA system into current or future Army command and control systems.

Section 2 of this report, supported by Appendices A and B, provides insight into the problem of developing an OMA system from both the G3 operations staff perspective and the technology application perspective. Section 3 presents the results of our considerations of mapping various technological constructs onto the operations monitoring domain, a design of an OMA system. This design is meant to be a starting point for building an OMA system in Phase II, using the typical AI expert system building paradigm of constructing a "bare-bones" prototype and then evolving it to a capable system by using it on increasingly complex simulated operations monitoring problems to refine and extend its design. The main body of the report concludes in Section 4, a short description of OMA system development plans. Appendix A presents an abstract scenario situation used in our study and Appendix B presents summary statements about the various components of AI planning technology.

2. OPERATIONS MONITORING ASSISTANT -- SYSTEMS DEVELOPMENT PROBLEM DESCRIPTION

The primary purpose of our research is to provide a design for an interactive operations monitoring assistant man-computer system for use at the corps level by the G3 and his staff. Figure 2-1 abstractly indicates the major functions of concern to operations monitoring. There are two major components of understanding of the OMA systems development problem on which our design will be based. The first is understanding the functional tasks that it must perform, or aid the G3 staff in performing. The second is collecting and understanding the technologies applicable to performing these functions. This chapter is divided into two major subsections that respectively provide insight into the problem from the military domain perspective and from the technology perspective.

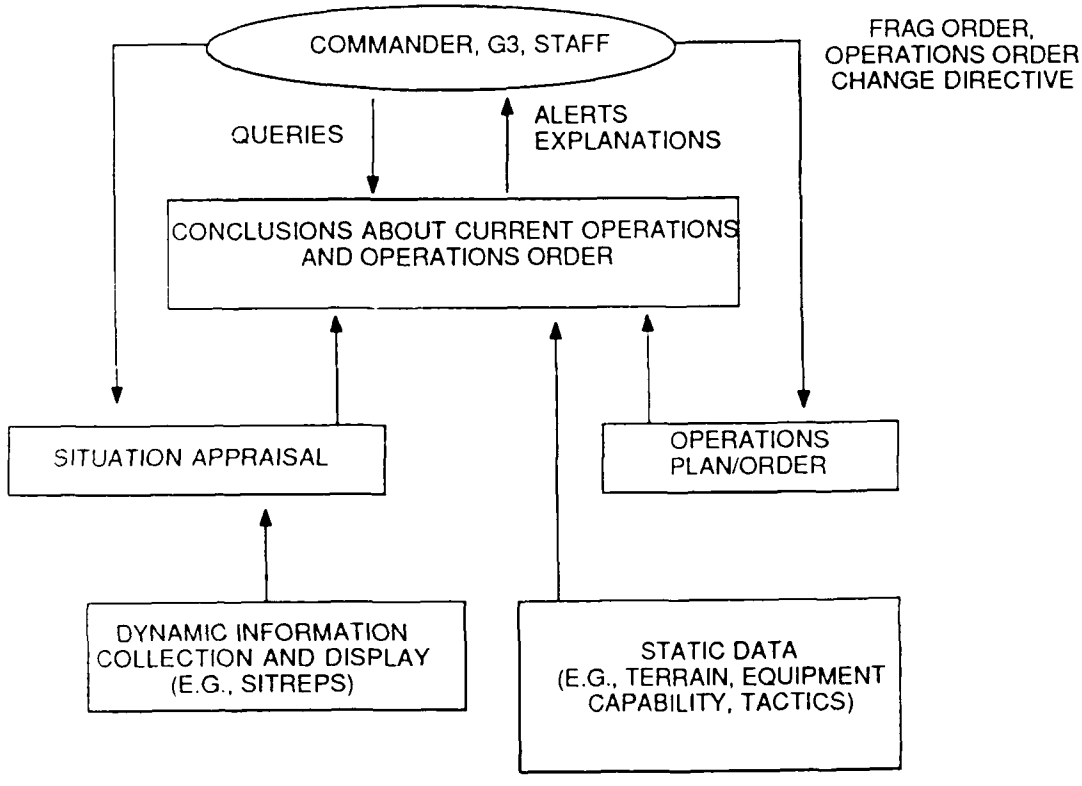


Figure 2-1: Operations Monitoring Concept

2.1 DOMAIN PERSPECTIVE OF OPERATIONS MONITORING

Operations monitoring by the Operations Division of the Corps G3 Operations Staff, abstractly stated, comparing the current corps' subordinate and supporting forces' operations with the planned operations for the purpose of identifying when changes in the planned operations of own forces should be made, as indicated in Figure 2-1. "Small" changes would be initiated by the Current Operations Division, while needs for more long term changes might be considered by the Plans Division.

2.1.1 Breadth of Operations Monitoring

In practice operations monitoring is an integral part of controlling the current operations of approximately 50,000 soldiers and their 15,000 vehicles of a variety of types, organized into a hierarchical Corps organization. These corps resources are supported by additional thousands of personnel and equipment (e.g., Tactical Air Force) and work in coordination with adjacent forces (e.g., other U.S or NATO corps). The corps may be in conflict with an adversary with a third up to three or more times the corp's resources. Thus corps level operations monitoring is necessarily comprised of a complex, multi-faceted set of activities and reasoning processes.

Major bodies of facts, knowledge and procedures used in monitoring and orchestrating coordinated operations of all friendly force units include those listed in Table 2-1.

2.1.2 Use of Planning Information in Operations Monitoring

The partial list in Table 2-1 indicates there is a myriad of concepts and details that the Operations staff must have organized in the forefront of their minds, or immediately available, to be effective in monitoring and controlling the corps resources. One of the most important requirements for the monitoring function is a thorough understanding of the planning factors that were considered in developing the current Operations Order, as indicated in Figure 2-2. In developing the Operations Order the Operations staff typically would consider the probable objectives, tactics, operations and style of command of units one and two levels below them (i.e., division and brigade) and any special supporting

Table 2-1: Operations Monitoring Factors

- EAC objectives
- EAC guidelines
- Corps mission and objectives
- Corps Commander's guidance
    - Assumptions
    - Constraints
    - Special instructions
- Assigned Corps resources
- Supporting forces (probably) available
- Missions, objectives and guidance assigned to next lower subordinate commands
- Principles of war
    - Objective
    - Offensive
    - Mass
    - Economy of force
    - Maneuver
    - Unity of command
    - Security
    - Surprise
    - Simplicity
- METT-T
    - Mission
    - Enemy
    - Terrain and weather
    - Troops available
    - Time

Table 2-1: Operations Monitoring Factors (cont.)

- COCOA
  - — Critical terrain features
  - — Obstacles Cover and concealment
  - — Observation and fields of fire
  - — Avenues of approach
- Typical TO&E for own forces and enemy forces
- Standard tactics, operations and unit capabilities (e.g., as contained in the scores of field manuals)
- Equipment characteristics
- Current estimates of actual TO&Es
- Current estimates of units' locations, activities and capabilities
- Current Operations Plan or Operations Order, current FRAGOs
- Superior, adjacent, subordinate and support Commanders' styles of operations
- Data and information received in formal reports (e.g., operational Situation Report, Periodic Intelligence Report, Periodic Logistics Report, Personnel Status Report)
- Data and information received via radio command and control channels
- Data and information attained through face to face meetings and conferences
- Estimation of engagement outcomes
- Estimation of resupply rates

units that may be expected to play a key role because of factors special to the current situation. Thus, they may consider, in detail, the outcome of what they perceive as likely engagements. However the purpose of this planning analysis is to ascertain the organic and supporting resources, and timing, they should assign to specific missions they specify to attain the corps' objectives. They will usually refrain from telling the subordinate and supporting commanders what tactics and activities to use in performing their assigned missions. However during execution they will expect that subordinate and supporting forces will report the types of operations and activities they are conducting, as well as their effectiveness and results. These reports combined with a thorough understanding of the plan enable the Operations staff to form expectations of future situations, activities and performance that expedite the assimilation and analysis of future reports.

Since many factors are likely to change between the time of planning and execution time, and because the subordinate commander will have more detail about his operational environment and his own personal style, many activities are likely to occur at division and brigade levels that were not explicitly planned or considered by the corps Operations staff.

Thorough planning analysis leads to a "working plan" that is only partially recorded on greaseboards, clipboards, maps, and computer files. The "working plan" is integrated in the Operations staff's heads in accordance with their understanding of the hierarchy of goals, constraints and plans; and the myriad of

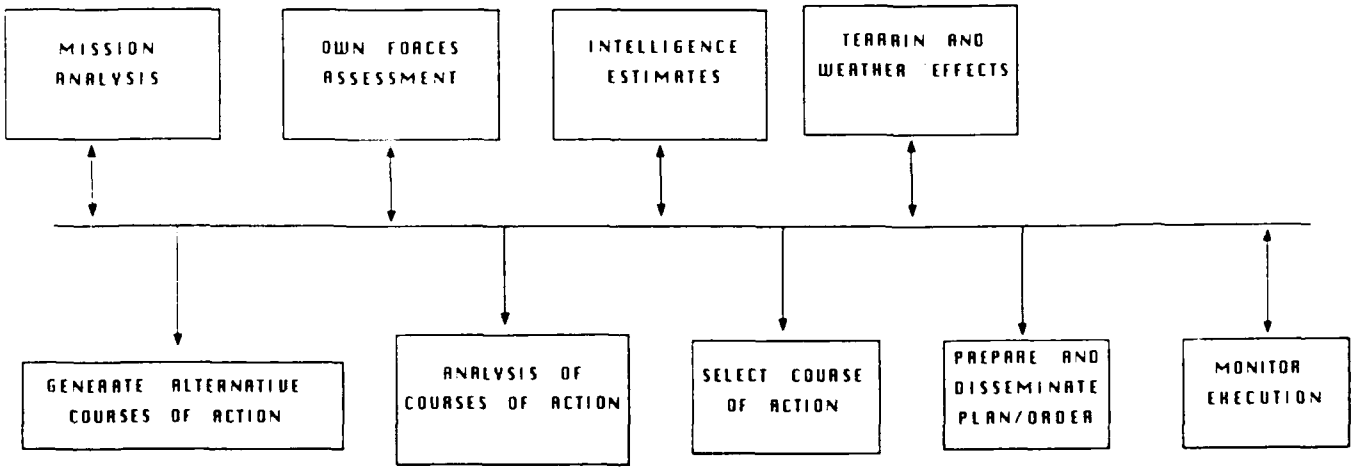


Figure 2-2: Planning Factors

planning factors considered in arriving at the current plan. The "working plan" needs to accommodate the US doctrine of flexibility and delegation of authority to the on-scene commander. Thus the "working plan" in the Corps G3 Operations Division must itself be flexible and adaptive to the own force, enemy, and natural environments; especially to the prerogatives of the subordinate commanders. Monitoring data, as it arrives, is assimilated and evaluated against this "working plan." The data and its implications need to be evaluated against such things as higher level goals, higher commanders' guidance and concepts of operations, principles of war, and high level models of combat engagements (e.g., heuristic models, Lanchester type models), rather than just ascertaining if an activity is "rigidly" adhering to a specific tactic or specified process.

The corps operations monitoring function is data driven; data is obtained from various reports, meetings, higher directives, etc. Since many "unexpected" operations are likely to occur, by both enemy and friendly subordinate units, operations monitoring staff need to deduce from the arriving data both what types of tactics and activities are being executed, and how effective they are in achieving the goals of the unit involved and those of the parent units up to EAC. For example, suppose a brigade "unexpectedly" reports to its division commander, with an information copy to corps, that it is crossing a river that has only one fordable position (which the brigade is using) to control a terrain strong point overlooking a potential enemy division size avenue of approach. The corps operations staff would perform their own evaluations to estimate how well this brigade operation supports the division objective, and in turn the corps and EAC objectives. They would consider the opportunities presented for division, corps and EAC; the risk that may accrue to the brigade, division, and corps; and any additional resources that may need to be assigned to either exploit opportunities or protect the involved units (i.e., whether they should alter their plan). If their analysis, whether heuristic or aided by various closed form or simulation models, indicates that the brigade activity is in support of all superior units' objectives and does not violate any constraints or the commander's guidance, then no alert or change of plans would be initiated. If, however, they estimate that the brigade is in little danger, and can easily block the enemy division's avenue of approach, and the adjacent advance friendly division is further forward than had been anticipated for this time, then they could decide to initiate a flanking maneuver against the enemy division anticipated to become stalled in the avenue of approach leading to the river crossing.

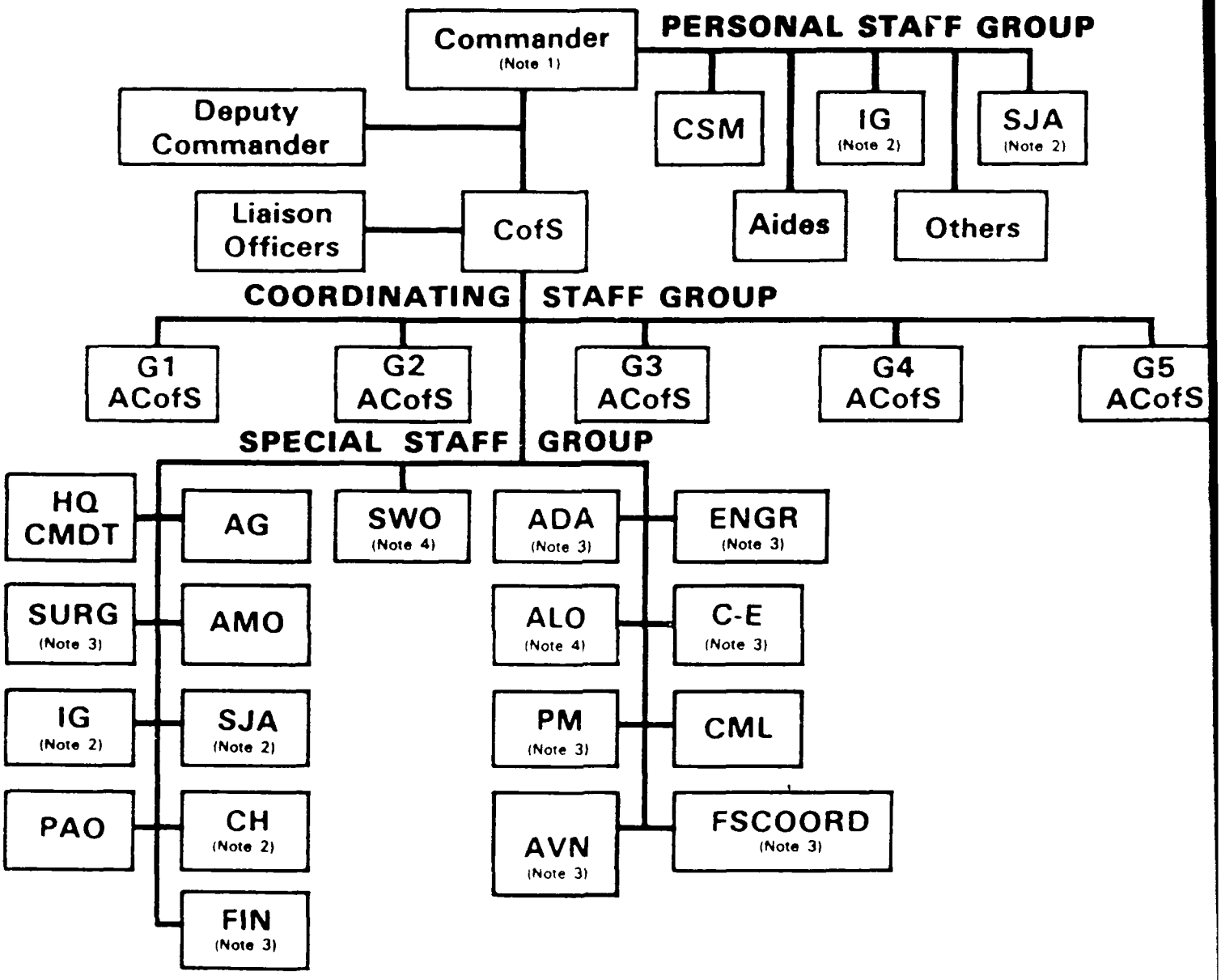
2.1.3 Data Flow and Analysis

Data for operations monitoring is contributed by large numbers of soldiers on the battlefield that channel their information into the G3 Operations staff and or directly to subordinate commanders and their staffs. In particular each part of the staff organization (e.g., G1, G2, G3, G4, G5, special staff organizations, officers (FSCOORD,ADA,AVN,ALO,C-E,ENGR)) (see Figure 2-3), subordinate commands and supporting forces are responsible for providing current status and activities information to the G3 Operations staff. Additionally higher commands and National intelligence organizations filter and provide information pertinent to operations monitoring in the corps area. All these sources may offer their information periodically or when they think it is probably needed, and respond to direct requests for additional information. Much of the tactical operations data comes into the G3 Operations staff via C2 radio circuits and personal visits. However a great deal of information also comes into the corps Headquarters staff areas to which it is functionally most pertinent, filtered by staff officers there and then elements of it communicated to G3 Operations.

Thus operations monitoring is a process that is continuously collecting and processing data on all aspects of the war within the corps area of responsibility, analyzing it for tactical implications and alerting the OPS duty officer when changes should be made or the need is anticipated, both for exploiting opportunities and for preventing undesirable effects that may be caused by the enemy or environmental events. The evaluation of tactical opportunities and risk situations must consider the effects of all perceived characteristics of the forces and environment involved, not just the major elements of maneuver and fire support. Factors such as arrival of key personnel at critical points on the battlefield, personnel fatigue, unit readiness, esprit, engineer's availability, and amounts of support material in supply (in addition to "mainline" items like munitions and POL) may in certain tactical situations be heavily weighted factors in operations monitoring decisions.

2.1.4 Simultaneous Evolution of Planning, Execution and Monitoring

A corps level plan is an evolving entity. It is basically a hierarchical assignment of resources to objectives for varying time intervals with explicitly stated and implied attendant constraints and guidelines. In many scenario



OTE 1: Special staff sections have been grouped under the coordinating staff section sponsible for primary staff coordination.

OTE 2: Direct access to the commander as a personal staff officer as required. The and the SJA, by regulation (AR 20-1 and AR 27-1), will be members of the personal group.

OTE 3: Also subordinate unit commander.

OTE 4: Provided by Air Force.

(SOURCE: FM 101-5)

Figure 2-3: Corps Staff -- Operations Monitoring Information Sources

situations of interest, part of a plan may be executed while other parts of the plan are still being formulated. The plan is not developed at a uniform level of detail in all its parts. Sometimes lower level parts of the plan, that are ahead in the planning execution of the overall plan, affect the evolving plan (in other parts) at higher levels of aggregation. Thus an overall common appreciation of the plan goals, guidelines and constraints are necessary for a good evolving plan, rather than just a rigid specification of resource allocation and associated deterministic parameter values and value intervals. The data driven monitoring system must at times interpret the meaning of arriving data in the context of an understanding of the goals, commander's guidance, constraints , and principles of war.

While a plan is being monitored, some of its components may be specified rather rigidly with definite parameters being required to be maintained within specified limits. The on-scene person in charge will need the freedom to interpret and infer appropriate parameter value intervals for other plan components. This interpretation and inferencing may be done purely heuristically(most commonly) or may include the use of plan evaluation aids (e.g., combat models). Thus constraint checking mechanisms for specified goals are being considered together with mechanisms for specifying "soft" goals, associated guidelines and constraints for sub-parts of the overall plan; all in consonance with the apriori common appreciation of the overall plan goals, guidelines and constraints. The evaluator should perform his evaluations from the perspectives of several echelon levels. Does the data indicate whether: 1) any of his subordinates are in trouble or have an opportunity for exploitation? 2) his own force echelon is in trouble or has an opportunity? 3) the next higher echelon is in trouble or has an opportunity?

Corps plans are monitored within the contexts of many different echelons, functions, and staff perspectives. Goals at all echelon levels are not precisely understood, as viewed from another level in the hierarchy. Prescribed activities to attain goals are purposely not sharply defined, leaving lower on-scene commanders flexibility to adjust to changes in the enemy, environment, and own forces as the changes occur. This flexibility also provides flexibility to each commander as to what he thinks should be reported to other commanders and among the various staffs.

2.1.5 Efficient Operations Monitoring Provides a Force Multiplier Effect

Operations monitoring is more complex than just deducing from data whether or not all units are successfully performing prescribed tactics and activities written in a specified plan. If it is done expertly and efficiently, then there is a greater potential for the friendly units to act inside of the enemy's "detect-decide-act" cycle, and thereby gain a force multiplier effect. For instance, in the example in Section 2.1.2 above the adjacent division can go immediately on the offensive (one of the principles of war) rather than maintaining the original plan of gaining it's originally assigned area objective and defending for an interval of time. That plan would have required yet another force to successfully engage the enemy division in the same time interval; the friendly brigade in defense against the advancing enemy division being judged a standoff (the attacker having an approximately 3:1 advantage). Experience, orderly knowledge, training and technological support are all needed to achieve the efficient, effective operations monitoring required.

2.1.6 Significant Desired Features for an OMA System

The material presented in the preceding subsections is based on studies of several Army Field Manuals, particularly FM 100-5 "Operations", and FM 101-5 "Staff Organization and Operations"; analysis of operations and a scenario compiled by the U.S. Army Command and General Staff College (CGSC) Reference 3, discussions of the abstract scenario in the OMA proposal (described in Appendix A), observations and discussions during the Crested Eagle Command Post Exercise (CPX) at Ft. Lewis, and interactions with several retired U.S. Army officers. The following features are deduced as being desirable in an Operations Monitoring support system (i.e., an Operations Monitoring Assistant (OMA)). The OMA:

1) Should be a mixed initiative soldier-machine system (i.e., a soldier-machine system in which either the soldier or the machine can lead the operations monitoring process, as the soldier desires).

2) Notes changes in status or activities previously planned or reported by subordinate or supporting forces and alerts the G3 of significant changes.

3) Does goals-constraints analysis when significant changes occur; and posts consequences on own and enemy forces (e.g., by using appropriate heuristic or operations research models of potential engagements).

4) Should identify opportunity and risk situations in a timely manner.

5) Provides explanations and justifications of its alerts, and of its opportunities and risks notifications.

6) Presents status information at various organizational and mission function levels.

7) Supports the "detect-decide" part of the "detect-decide-act" cycle in a timely manner.

2.2 TECHNOLOGY PERSPECTIVE OF OPERATIONS MONITORING

The field of operations research (OR) has long studied and contributed to technical support of command and control of tactical forces resulting in a number of tools including:

- Sensor capability models
- Resource assignment algorithms
- Queuing models
- Route selection procedures
- Movement models

- Terrain masking calculations
- Weapons effects models
- Logistics models
- Communication network models
- Combat engagement models
    - Force ratio guidelines
    - Lanchester equations
    - Monte Carlo simulation models such as CORDIVEM and CORBAN
- Statistical analysis packages

These OR tools are useful in planning operations, but typically require expert operations analysts to use them individually, and as a set. These types of tools continue to be developed and refined for planning of tactical operations. Most of them are not used during operations; (that is, for operations monitoring).

This study recognizes the importance and utility of such OR tools but concentrates on the technologies within the AI field. However it is the intent of the OMA system design in Section 3 to use the most pertinent forms of knowledge to perform the operations monitoring functions; a knowledge-based system (rather than a (single) expert system) is described that makes use of OR, AI and human forms of knowledge. This section does not explore OR or human analysis and reasoning explicitly, but concentrates on AI. AI techniques can be used to control, run and interpret OR models, and to call on the soldier's intellect when needed.

The soldier-machine OMA system supporting the command and control functions described in Section 2.1 above must perform the following functions:

- Collect, filter, collate and display data from sources.
- Indicate deviations from plans.

- Alert the decision maker to opportunities and risks.
- Justify the alert.
- Evaluate (partial) plans.
- Justify and explain the evaluation.
- Provide basis for formulating and disseminating a FRAGO.

AI techniques may be applied to aid in performing each of these functions.

Most of the preceding functions have been addressed in the AI area of planning, but only to a modest degree, and little has been done directly in the area of plan execution monitoring, (i.e., operations monitoring). Appendix B presents an overview of AI planning technology that serves as a basis for the OMA system design in Section 3; the reader is encouraged to read the Appendix now since most of the pertinent technology overview is there rather than in this short subsection. Of particular importance to operations monitoring are plan representation, situation representation and reasoning about their differences and implications. Reference 4 presents a group of papers and survey on knowledge representation issues that also provide important insights for the OMA system design.

Brachman and Levesque's bibliography survey 4 characterizes the various methods of representation into:

- Procedural representations
- Formal logic-based representations
- Structured object representations (frames)
- Associational representations (networks)
- Other representations

"Other" includes use of more than one of the previously listed types in a system. We believe the above list should be extended to include model-based

representations such as used in the OR models previously mentioned and also being developed by AI researchers for applications such as diagnosis of equipment and systems (e.g., see Reference 5).

The hardware technology for an OMA system is not addressed in this Phase I feasibility design study. The correct application of state of the art software techniques is the major concern in designing and building an OMA system. Several hardware systems exist which can adequately support the OMA software.

2.2.1 Mixed-Initiative System Technology for OMA

AI systems usually work in one of four interaction modes with a human:

- Autonomous (e.g., robot) -- the human prescribes task and turns the machine on; the machine does the task.
- Consultation (e.g., medical diagnosis and treatment) -- the human supplies data about subject and tests; the machine does the diagnosis and suggests the treatment.
- Partitioned tasks (e.g., image interpreter workstation) -- the human does what he does best, such as complex pattern recognition, spatial relationships; the machine does tasks it does best, such as segmentation, mensuration, filing data, cross referencing.
- Mixed-initiative (e.g., operator aiding systems) -- the human and machine jointly reason and control, with one or the other, performing tasks as the operator's work load, focus of attention and desires dictate.

The desired features listed at the end of Section 2.1 clearly indicate that the OMA system should be a mixed-initiative system.

It is also clear that the system should employ multiple types of representation: frames for status of hierarchically organized units, and for hierarchical mission goals assigned to the units; procedural knowledge for such things as checking constraints, prioritizing the order of frame slot filling, and deciding among candidate slot fillers; a mixture of procedural networks and

frames for characterizing sequences of actions of units and interrelationships among these actions and units; and model-based reasoning for estimating outcomes of potential engagements.

Although no substantive examples of monitoring systems for complex operations have been built and demonstrated, the individual technologies have been developed for plan representation; situation and activity representation; data collection, collation and display; recognizing significant events that can be characterized in terms of specified characterization features of the objects of interest (e.g., own force units, enemy units, terrain, weather); deducing potential consequents of significant events; alerting soldiers; explaining and justifying evaluation conclusions; and accepting interruptions and re-direction from the soldier. Rather than discussing these technologies somewhat generically in this section, we will provide needed insight in the following OMA system design section.

2.2.2 OMA System Development Environment Technology

The OMA system when completely developed and fielded will need to fit into the hardware, software and communications environment at that time. It will need to interact closely with the planning system and command and control systems, such as (perhaps) the Maneuver Control System. However, to demonstrate feasibility the emphasis should now be on developing the system in a good development environment. A machine, such as Symbolics or SUN-III that robustly supports Common LISP software development, should be used. Assumptions that plan and monitoring data that can be communicated easily into the OMA system data bases should be made; although some considerations must be made for accomodating protocols and formats of data in current and future status report and C^2 systems, DMA map and feature data, and other static data bases.

The specific software environment that should be chosen for building what will become a complex knowledge based (KB) system is more complex. Several expert system building environments, or shells, have been developed and are being highly touted by the companies that sell them, and many government program managers alike. However, there is controversy about the applicability of the current (but evolving) system to the development of large complex AI

systems. The following quote from the Erman, Lark and Hayes-Roth paper Reference 6, "Engineering Intelligent Systems: Progress Report on "ABE,"[1] summarizes some problems with using these current tools:

"Most people now perceive a gap between what the intelligent systems technology should be able to do and what can be done today. While the technology holds great promise, it cannot yet supply solutions readily for many of the problems for which it should be applicable. Today, that technology transfers from research environments to applications chiefly through *knowledge engineering tools*. Prominent examples of these are the commercial products ART (from Inference Corp.), KEE (from Intellicorp), KnowledgeCraft (from Carnegie Group), and S.1 (from Teknowledge). These tools incorporate the best methods of applied artificial intelligence, and they reflect some of the best techniques for building expert systems. However, these tools currently have several weaknesses. Generally, they reflect the small-scale and isolated nature of the applications that motivated the tools. Specifically, the major problems include the following:

- The best current tools are monolithic, single-purpose software packages. Hence they are hard to extend or apply beyond their current range of applications. They are also difficult to integrate with conventional data processing and computer technologies.
- The tools provide capabilities that are low-level. Most applications require the user to build a solution structure on top of those primitive capabilities. This design and implementation work is expensive and time-consuming, and requires a skilled and experienced knowledge engineer.
- The tools support a limited variety of data types and inference schemes.
- The inference schemes in current tools are built-in and practically hard-wired.
- Current tools do not support large-scale applications.

---

1 ABE is a trademark of Teknowledge, Inc.

- The tools have been designed exclusively for uniprocessor implementations.
- The tools have not been designed in a way that makes them easy to port to alternative new machines."

Because of problems such as these we have found at ADS that expert programmers frequently drop out of the shell into the more flexible LISP or C environments when the shell constructs or inferencing mechanisms don't directly provide the required capability. This helps to quickly achieve a working feasibility model with most of the desired features included in a "bare-bones" way. However, whether less expert knowledge engineers (as distinguished from expert AI computer scientists) can efficiently and accurately continue the growth of the system into a robust, large KB system without higher level tools, such as those the tool manufacturers are trying to produce, is still an open issue. There will be trade-off decisions to be made between the fidelity of the model produced, and the ease of doing the development work. The decision to use a high level tool such as the ones mentioned in the quote above an advanced tool, or stay with, say, Common LISP, should be made at the time the system build starts.

3. OMA SYSTEM DESIGN

The operations monitoring concept was discussed in Section 2.1 and abstractly represented in Figure 2-1. This section examines that concept in more detail for the purpose of providing a conceptual design of a corps level mixed-initiative OMA system. At the technical heart of such a system there must be workable representations of the current plan, and the current and expected situations; and reasoning mechanisms to recognize from incoming data deviations from the current known plan, and to also identify opportunity and risk situations that were not necessarily anticipated or contained in the current plan. Additionally there must be data input capabilities and efficient soldier-machine interaction capabilities. (Note: We believe that with a successful soldier-computer OMA system, it will still be desirable for many years to come to have the system embedded in the maps, overlays, clipboards and greaseboard status displays environment used now). Following subsections address these system design issues in turn.

3.1 OMA SYSTEM OVERVIEW

Figure 3-1 presents the concept in Figure 2-1 in more detail, from an OMA system's perspective rather than the military functions perspective of Section 2.1. We assume that the OMA system will be closely associated with the corps plan generation and evaluation (PG&E) system when they are all developed. In particular, we assume the formal OP PLAN or OP ORDER and FRAG orders will be available and that the plan generation and evaluation Knowledge Sources (KSs) and engagement effectiveness models and their data bases that were used in the planning process, will be available for use by the OMA system. If they are not available and easy to use during the Phase II effort we will rely on the soldier to generate appropriate plan elements and the soldier-machine system to use high-level abstract evaluation schemes of proposed new plan elements. To further focus on the OMA problem we also assume that on-line input monitoring data will be available in the correct formats and that (prompted) manual input of other required data will be acceptable; at least through our Phase II development of an OMA system. Thus, for our Phase II feasibility OMA system model, we will concentrate primarily on the internal representations and reasoning

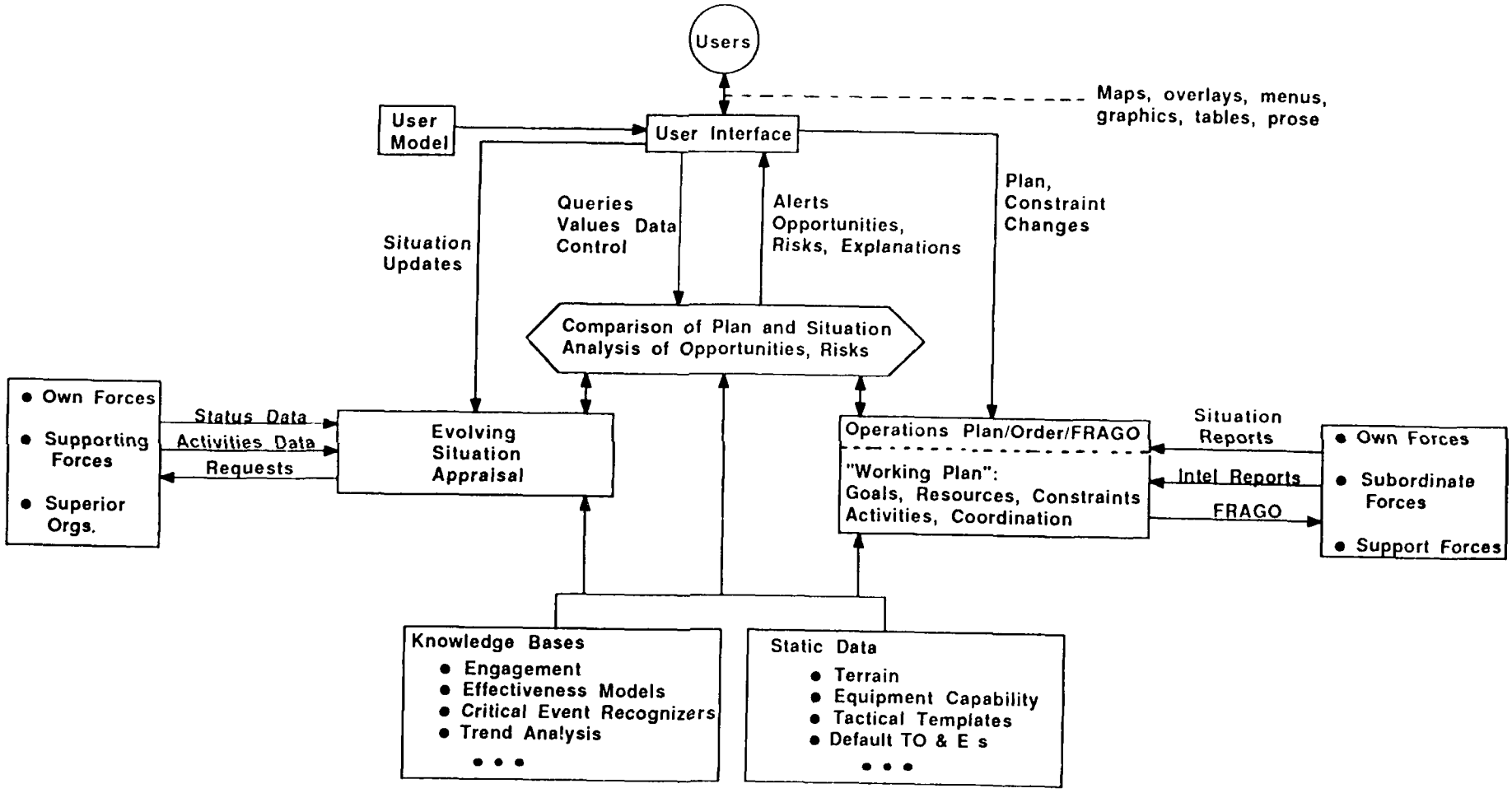


Figure 3-1: OMA System Overview

mechanisms required for an OMA system.

The user, who can be any of the staff officers, will interact with the OMA system through the user interface. This user interface will probably consist of two CRT's, one color and one black and white. The color will have terrain and feature overlay capabilities, as well as alphanumeric displays. The black and white will be used primarily for input-output via menus, tables, graphics, and so forth. In later, mature models of the OMA system, user models will be provided so that when a staff officer, say the G4, is using the system, the user model will more efficiently provide information of interest to logistics considerations. The user can enter changes to plans, FRAG orders, changes in the constraints, or concept of operations, situation update information and system control directives. The user interface will direct this type of data and control instructions to the proper areas and files within the OMA system.

There are two major dynamic data base areas within the system. One is to keep a current evolving situation appraisal; the other is a current formal operations order and FRAG order and also a current working plan that the soldiers and machine most frequently use in monitoring the operations.

There are two major types of reasoning continually being conducted in the system. One is comparing the current evolving situation appraisal with the current operations order or FRAG order to determine whether the soldier and planning system should be alerted for re-planning. The second analysis area is for identifying situations in the battle environment that provide opportunities for own forces and also situations that provide opportunities to the enemy and risk to own forces. Explanations and justifications will also have to be generated for any alert or notification of opportunities and risks that are given to the soldier.

The data bases and reasoning within these three major areas of the OMA system are supported by static data that does not change frequently during the course of the corps operation and various knowledge bases that can be used in analyzing the data and potential plans for responding to opportunities and risks.

In a tactical options generations study previously done at ADS 7 it was concluded that the decision maker and his staff typically do several functions at essentially the same time with frequent mental and procedural jumping from one type of analysis to another. Figure 3-2 indicates the type of analysis that the

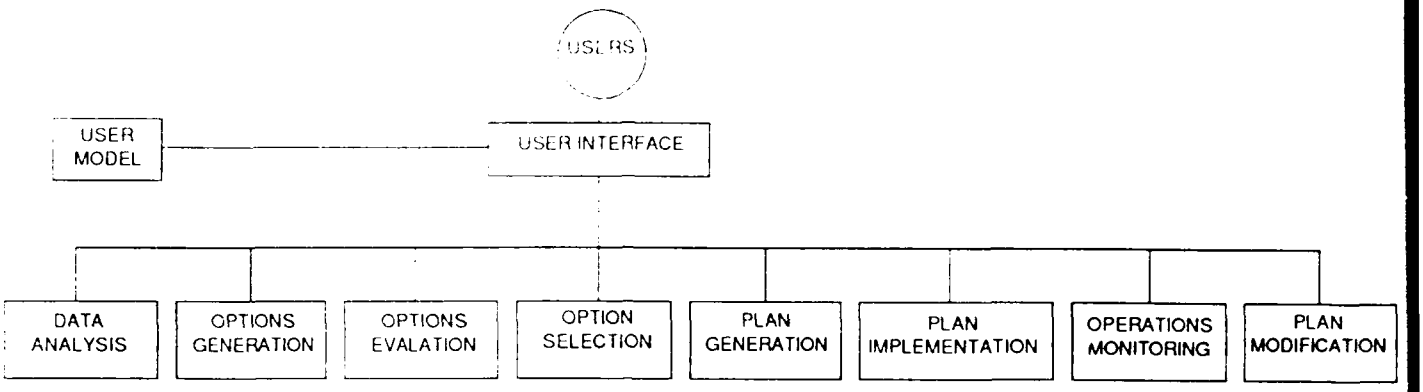


Figure 3-2: Planning and Monitoring
Functional Environment

decision maker and its staff performed in this "helter-skelter" manner. The last two boxes on the right, monitoring and plan modification, have been added for this report. Note that the order from left to right of these boxes is the same order as recommended in making the commander's estimate of the situation. The decision maker does these activities in a cyclical way to start with, from left to right; for easy problems, that is sufficient. For more complex decision problems, it appears that he and the staff need to get a deep working understanding of all elements of the problem area from data analysis through plan implementation considerations. We feel that this environment will have to be extended to include the last two areas of analysis in order to be able to do the operations monitoring job correctly in difficult situations.

3.2 PLAN REPRESENTATION

3.2.1 OP PLAN, OP ORDER, FRAG ORDER Representations

The operations plan and operations order have a definite format with definite content for each paragraph and a definite order for the annexes that support the plan with rationale and specific data. The OMA system will contain a verbatim copy of the OP PLAN and/or OP ORDER. An efficient editor will be included to enable finding any part of the plan or type of data efficiently. When changes are issued, the editor can also be used to edit the system's stored plan.

Similarly, FRAG orders will also be written into the system verbatim, however, since these FRAG orders are not necessarily well formatted, a soldier will interact with the system to enter information and change the working plan to reflect the FRAG order directive.

The OOB and other descriptive data contained in the OP ORDER annexes and arriving FRAG orders will be edited and transferred into the appropriate files within the static and dynamic knowledge and data bases of the OMA system. Additionally specific constraints inferred from the Commander's guidance and other pertinent directives will be extracted and specifically instantiated into rules attached to the appropriate pertinent slots within the plan, situation, and activities representations. If generic rules that can be modified to represent the current orders are not available, then a rule can be invoked to notify the soldier when the pertinent slot is accessed later during operations monitoring, if the soldier so indicates his desire.

3.2.2 Working Plan Representation

The operations monitoring staff and OMA computer system will need to spend considerable effort getting the formal operations plan into working plan form so that when operational data arrives they can quickly perform the required analysis and deductions. We propose filling in the working plan in the format indicated in Figure 3-3. The basic concept indicated in the figure is that at any point in time the corps resources are all engaged in a set of activities for specific purposes and that at a later time they should be engaged in other activities to

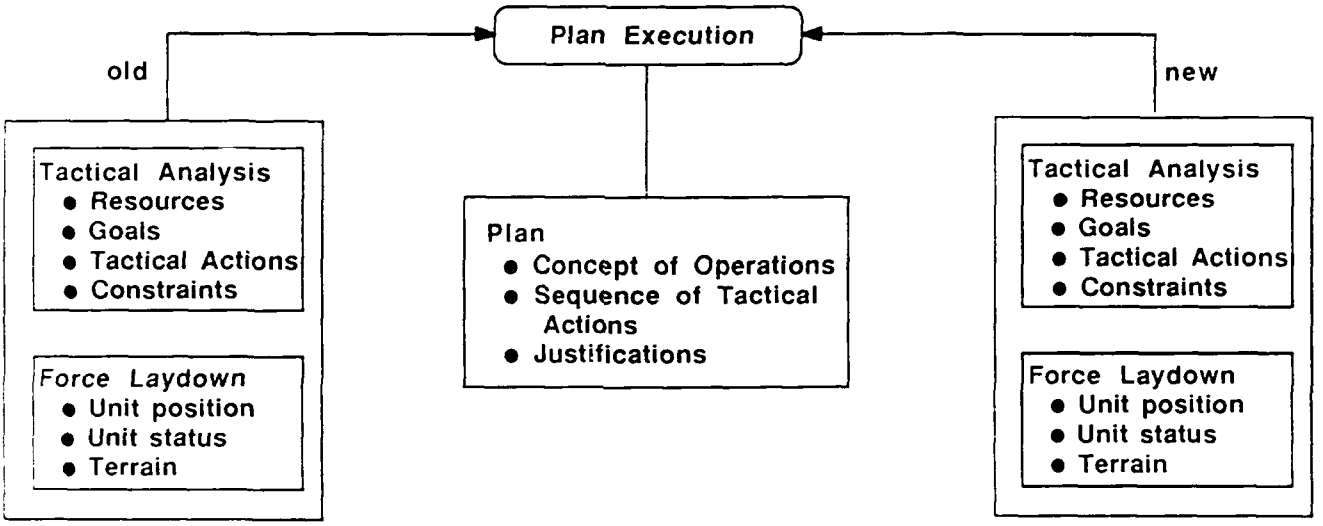


Figure 3-3: Plan: A Statement of Activities for Changing Tactical Situations

achieve new goals. Thus, the plan is simply the delineation of a sequence of activities according to some concept of operations, that will transform the state of the corps entities and their current activities into the desired state and the desired activities to attain the new desired goals. Providing justifications for the choice of the new goals and the choice of the sequence of tactical actions that will transform the state of the corps from where it is to the new desired state is needed for the monitoring process, as this transformation takes place. In Appendix A we have written up a fairly high-level description of the corps-level scenario that we've used in considering the issues leading to this OMA system design. Figure 3-4 represents a graphical depiction of a plan with justifications presented on the overlay to show the purpose of each new activity for each of the major units within the corps. We intend to develop concurrently these types of graphical depictions of plans together with internal computer representations of plans.

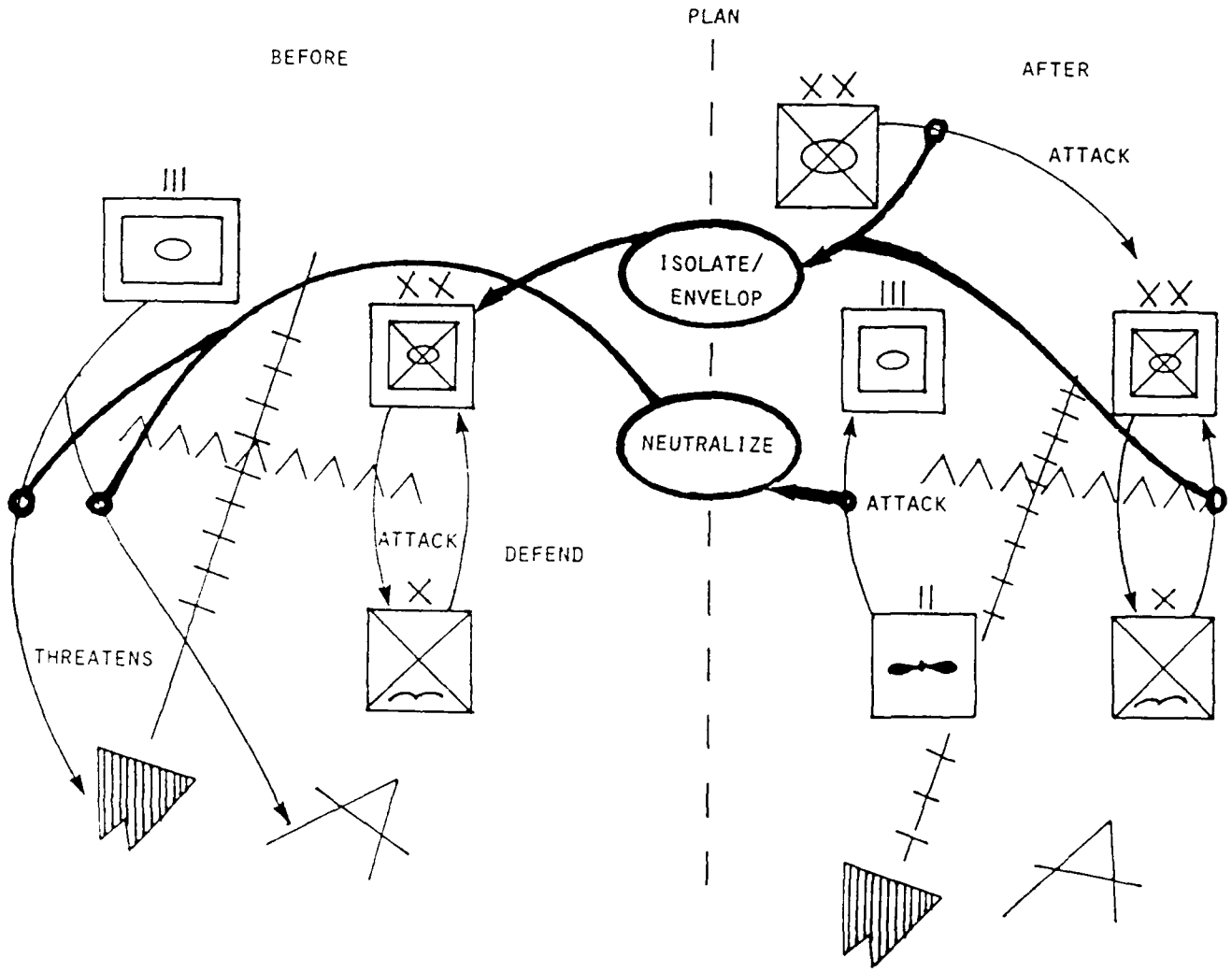


Figure 3-4: Justifications In a Corps Plan

A great deal of knowledge acquisition and knowledge engineering is required to develop the specific formats for all of the types of resources in the corps and its supporting forces, and all of the concepts of operations, goals, tactical actions, and types of constraints and justifications that are attendant to corps operations. Much of the data and reasoning needed are contained in the set of Army field manuals available, particularly FM-100-5, "Operations", 8, and FM-100-101-5, "Staff Organization and Operations". Additionally, expert Army planners and operations personnel will be needed to serve as domain experts. The focus in the Phase II feasibility demonstration system will be on maneuver issues, with small amounts of efforts in the areas of fire support, intelligence-EW, combat service support, and air defense.

More insight into this working plan representation, with some examples and some detail are given in a later section, following a discussion of hierarchical plan execution monitoring.

There will be numerous frame templates for various entities in the corps and activity networks for the various tactical actions that can be performed. A library of default templates and tactical action procedural networks will be developed and specified for the particular plan situations before or as operations monitoring begins. This specifying of actual values to be used in the frames and networks will be an interactive process with the soldier providing many of the data entries, and the computer system providing consistency checks and other types of constraint checking.

A large amount of the constraint checking will be procedural in nature: but they will be attached to particular slots on the frames, or positions in the network, and so there will not be a great number to search for any one slot filling application. This should keep the search time down and the efficiency of the system up.

It is important to do this preconditioning of the operations monitoring system before operations monitoring actually begins for particular objectives for two reasons: 1) To get the working plan up front in the soldier's head, 2) To provide as quantified a plan as possible against which to measure and asses deviations from the plan when operations status and activities data arrives. The initialization exercise will also indicate to the soldier, and perhaps to the machine system, weaknesses and potential opportunities that were noticed in filling out

the working plan. This sets the stage for recognizing opportunities to exploit by own forces and potential risk situations to be particularly watched for during operations execution.

3.3 SITUATION APPRAISAL REPRESENTATIONS

By situation appraisal we mean the consideration of all factors of own forces, the enemy, and the environment (e.g., terrain, weather). Thus, it incorporates the G2's estimate of the enemy, terrain and weather; but also all dimensions of own forces, especially those dimensions that affect their capabilities to perform the activities being conducted to achieve their objectives within the plan's stated time intervals.

The basic representation for a situation is the same as that described in the previous section for the "old" part of a plan (See the left side of Figure 3-3): a tactical analysis procedure and representation for the tactical analysis, including units or resources, their goals and tactical actions, and current constraints; and a force laydown description, including units' positions and status, and terrain information.

It is particularly important to know when a unit is changing, or is about to change, from an old set of goals and activities to a new situation. Hence, the situation representation will have special slots to so indicate, with demons attached to "look for" the specific types of data and indicators expected to become available when changes are made. For example, if a SITREP indicates that an enemy artillery unit just behind an enemy armored division, which has been occupying defensive positions in front of a friendly mechanized brigade for 12 hours, has initiated laying down heavy fires on the two forward friendly battalions in the brigade, then a situation representation special slot for indicating a major change in enemy activity at the division level would be filled with the activity identifier. Demons would be activated to monitor new data for other attack indicators, such as increased enemy close air support or enemy helicopter gunship activity against the friendly brigade and its supporting artillery. Demons for other situation activity states to which the enemy may be transitioning might also be invoked. Multiple potential new situations would be represented until sufficient confirming or dis-confirming data is accrued and interpreted to reduce the possibilities back down to one.

We will have default tactical action templates and activity networks, as well as various friendly and enemy force unit templates, all described in later subsections. These serve as representations for sub-parts of the situation appraisal.

Symbolic representations for terrain strong points that, for example, provide good observation points or positions with good fields of fire over potential enemy avenues of approach will be taken from other ADS terrain analysis efforts. These representations of terrain strong points will be linked, via pointers, to enemy and friendly force units near them, that is, those units whose areas of responsibility or influence contain the terrain strong points. Avenues of approach for various size units and trafficability factors will also be represented as they are in DMA terrain and terrain feature data bases and in other ADS terrain analysis efforts.

An important part of unit status information is that which describes unit readiness and capability. Each friendly unit will report its readiness, and the G2 will occasionally provide similar information about enemy units. This will contain a C1, C2, C3, or C4 unit rating and perhaps a break-out of the rating for the units' personnel, training, equipment and supply, especially if requested. The default tactical action templates will contain the types of activities each type unit is expected to be able to perform. These default values will be used to instantiate the various capabilities a specific unit possesses. As SITREPS arrive that indicate degradation in equipment and supplies, numbers of personnel, or loss of specific trained personnel, the units' readiness and capabilities for specific activities will change. A current capabilities table will be maintained for each unit for activities for which the capabilities values have changed from the default values, such as indicated in Table 3-1. This table indicates that the 13 Armored Division is capable of maneuver, but is no longer manned sufficiently well enough to have a good capability for offense operations. Its lack of (engineering) equipment makes it a poor candidate for crossing rivers.

3.4 HIERARCHICAL PLAN EXECUTION MONITORING

Operations monitoring compares the evolving plans and situations, as symbolically represented above, to find opportunities to exploit and risks to avoid.

Table 3-1: Capabilities Table

UNIT	READINESS FACTOR	CAPABILITY		
		Defense	Attack	River Crossing
13 Arm. Div.	personnel	good (C2)	low	good
	training	good (C2)	good	good
	equipment	good (C2)	good	low
	supply	good (C2)	good	good
	Overall	good (C2)	low	low

The corps is a hierarchical organization, as was indicated in Section 2.1, and the missions assigned to the corps typically have goals and objectives that can be divided into a hierarchy of goals and objectives for subordinate units and supporting units to achieve. Thus, it is necessary for a planning system and an operations monitoring system to reflect this dual hierarchical structure in organization and mission. Figure 3-5 is an abstract representation of a theory of hierarchical planning and hierarchical plan execution monitoring.

At the Nth level (for example, corps level) a superior echelon commander has provided the high level plan for the Nth level commander to carry out with his forces and supporting forces assigned. This Nth level commander has responsibility to assess the feasibility of the plan and mission provided him from his superior. He develops his mission plan into a more detailed plan that he feels appropriately uses his organic and supporting forces, and evaluates this expanded plan as to feasibility in terms of available resources expended, amount of resources and personnel potentially lost, and timing. He feeds the expected performance back up the chain of command and if performance results are below acceptable thresholds, then he should request more resources, or a change in objectives. Of course, he cannot make such recommendations intelligently without knowing the reference value system in which his superior commander is viewing the system. Thus, communications are required to have a common understanding of the value systems and the weighting among the various

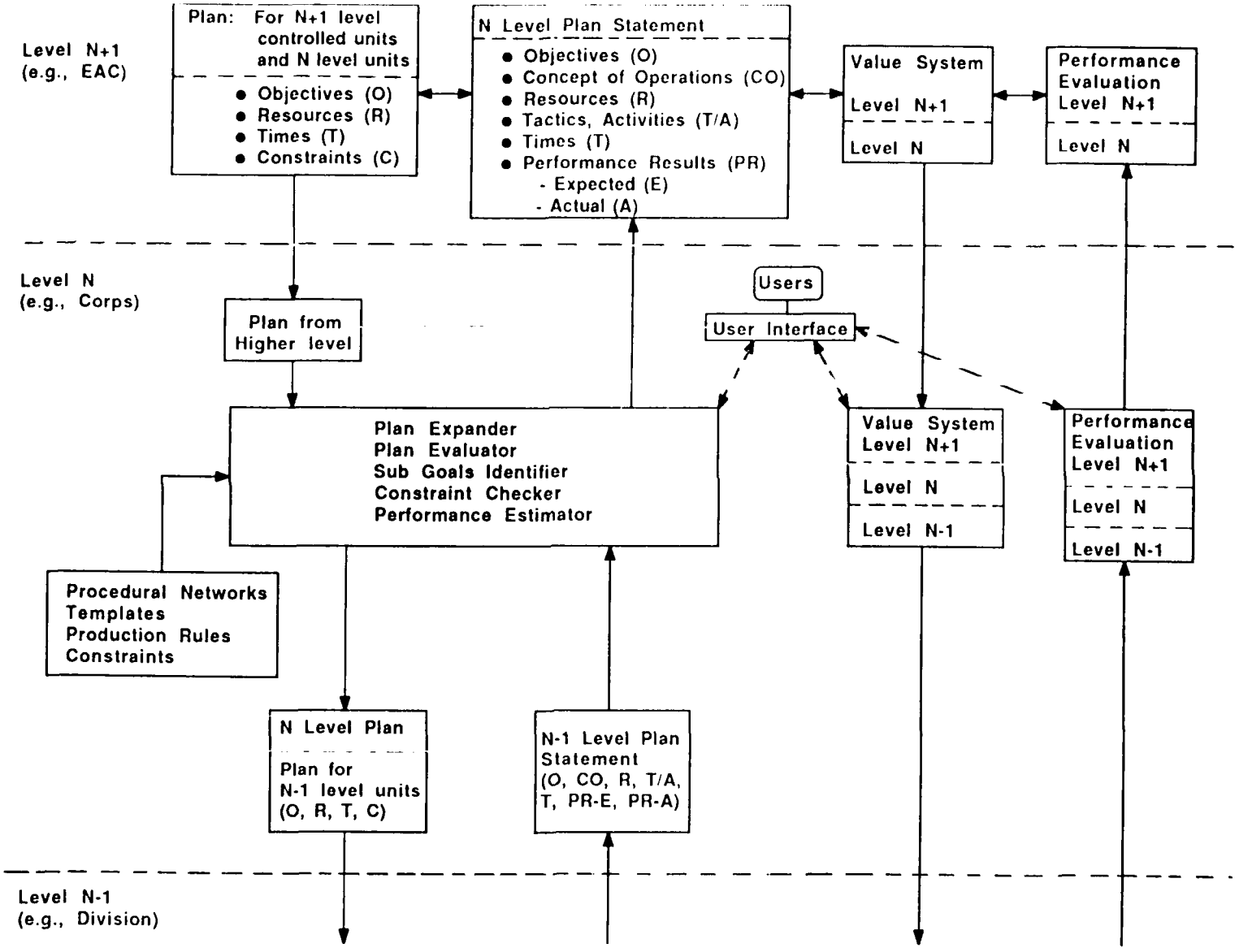


Figure 3-5: Hierarchical Plan Execution Monitoring

components within the value system. The same type of planning interaction is required between the Level N commander and staff and the level below him (the Level N-1 commander and staff).

The Level N commander and staff, in preparing their expected performance results estimate, need to look at the results from at least three perspectives: their superior and his mission, their own mission, and the level below them and potentially two levels below them, since, in doing their performance estimates, they may study engagement analyses using tokens for force sizes two echelons below them. They use these plan generation and plan evaluation tools to help determine the objectives and resources for the subordinate commands. But they do not tell their subordinate commands explicitly how to attain the goals assigned them. The subordinate command may find a better set of tactics and activities to attain the goals and should report their plan backup. However, this different set of activities may not also support the superior echelon's objectives and concept of operations as well as the one used in determining the objectives for the lower echelon commander, hence, the commander at each echelon level must view his activities in light of his own objectives, his superior's objectives, and perhaps the superior above that.

After operations start, operations monitoring becomes very similar to the planning process, but instead of using the performance estimator models and constraint checking mechanisms against expected situations, they are used for actual situations occurring or perceived as occurring. Again, each level commander and his staff should evaluate performance of the ongoing operations in light of his perception of how his superior commanders would view the activities for their own objectives and the constraints within which they work.

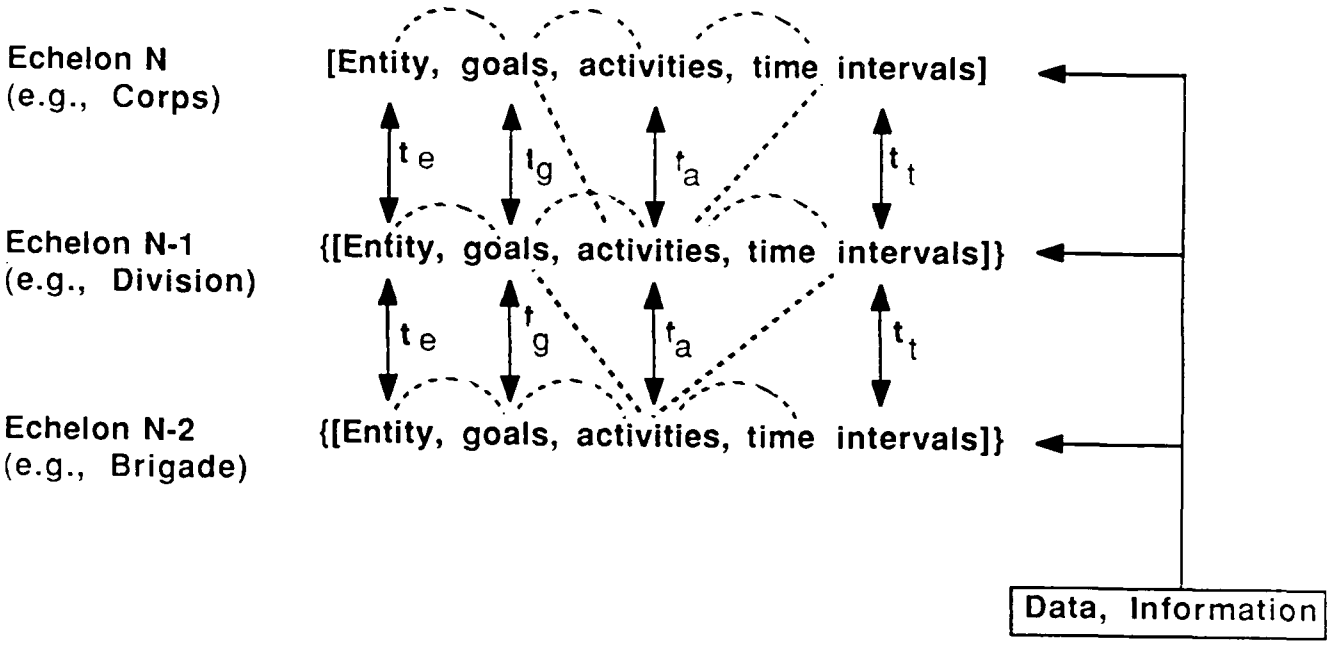
There is an assumption in this theory that there is cooperation and coordination among the hierarchy of forces in pursuing their hierarchy of objectives with the supporting resources assigned. Since each level commander is not likely to tell his subordinate commanders explicitly how to achieve their assigned objectives with their assigned resources, it is incumbent on each of them to keep communicating up the actual status of their unit's capabilities and existing resources, and also a statement of the activities they are performing and for which goals they are performing them. This type of information is required for the higher level commander to coordinate among the various forces and assure

that an accumulation of weaknesses does not exceed a risk threshold. Similarly, opportunities to exploit may be found by looking across subordinate commander's areas of responsibility and influence.

To use this theory, it is necessary to define hierarchical structures and symbols within the structure to permit describing the various plans and execution of the plan; and the monitoring of the plans as data from the operations comes into the monitoring system. Figure 3-6 suggests a symbolic *language* and relationships between the various types of elements represented by the symbols for the internal working plan representation. The entities in that figure refer to military units, and for coordinated operations the units have to have compatible goals and compatible activities during common time intervals of performing those activities. Thus, there are certain consistency requirements for a plan to be good among the entities, goals, activities and time intervals, as indicated by the dashed lines in the figure. This is across goals and activities at one echelon level, as well as between goals and activities and time intervals at multiple echelon levels.

Because of the hierarchical nature of both the organization and the missions, there is usually a "natural" decomposition into the next lower echelon set of resources, goals, activities and time intervals.

In operations monitoring there may be times when a subordinate, or when a subordinate's subordinate, does activities for assigned goals that his superiors were not expecting him to do. If this unexpected activity is not reported in the cooperative mode mentioned above, then the operations monitoring system can be expected to receive data from which it may be deduced (using forward chaining-like reasoning) that an "unplanned" activity is being performed, but that the activity is in pursuit of the assigned goals. Nevertheless, it has to be checked to be still consistent with all of the other goals at superior levels and other activities that are coordinating in the same time intervals. (This process of finding whether selected goals are supported by the data will employ backward chaining-like reasoning.) If the operations monitoring system is having difficulty in assessing why the activity is going on, or what it's implications may be if other levels are within the hierarchy, then it may, of course, request explanations from the lower echelon commanders that are performing the activity. However, a response to such a request may, at times, take a considerable amount of time and



Legend:

denotes consistency requirements

denotes transformation decomposing factor in hierarchical plan

denotes transformation composing factors in monitoring (check for completeness)

Figure 3-6: Symbolic *Language* Elements and Pointers for Internal Plan and Situation Representations

so as much of the analysis as possible should be done in the passive mode (that is, not putting out a signal requesting more information).

The language elements indicated in Figure 3-6, together with a structured object-oriented programming environment, such as SOPE 10, developed at ADS, should provide an efficient way of representing and storing plans and monitoring their execution.

Monitoring will be driven by data being reported into the G3 and by keeping track of the beginning and end times of all time intervals recorded in the plan. Major force status changes will be reported, and the projected consequences of those changes noted. The important category of mission completion will, in particular, be reported to the soldier. If a report indicates any unit in the hierarchy has completed a planned activity, the system will alert the soldier and also indicate other related units and activities impacted. If another activity for a unit is reported or inferred by the OMA system, then deductions will be made whether the planned activity was completed, aborted, or ever commenced; and the results presented to the soldier, again with a list of related units and activities. When activities to attain a major objective of any unit are recognized as being successfully completed, the soldier will be notified that that mission is completed. When the missions of all subordinate units are recorded as completed, the mission of the parent unit will be checked for completeness and its state of completion reported to the soldier.

3.5 HIERARCHICAL ACTIVITY NETWORKS

The hierarchical plan representation indicated in the previous section can be used by concentrating on any one of the four elements: entity, goals, activities, time intervals. Figure 3-7 indicates a scenario situation and a plan representation for corps level activity. By concentrating on activity we have easily come up with a representation that lends itself very well to graphical display of force planned activities. Time interval information is left off in Figure 3-7, but it could be easily added in as an alphanumeric entity or activity symbols. This hierarchical decomposition from corps to its three divisions to their brigades provides an example of how the soldier can interface through well-known icons, say, on the color CRT, as an overlay to a map background. This will facilitate time and distance considerations, provide quick access to terrain

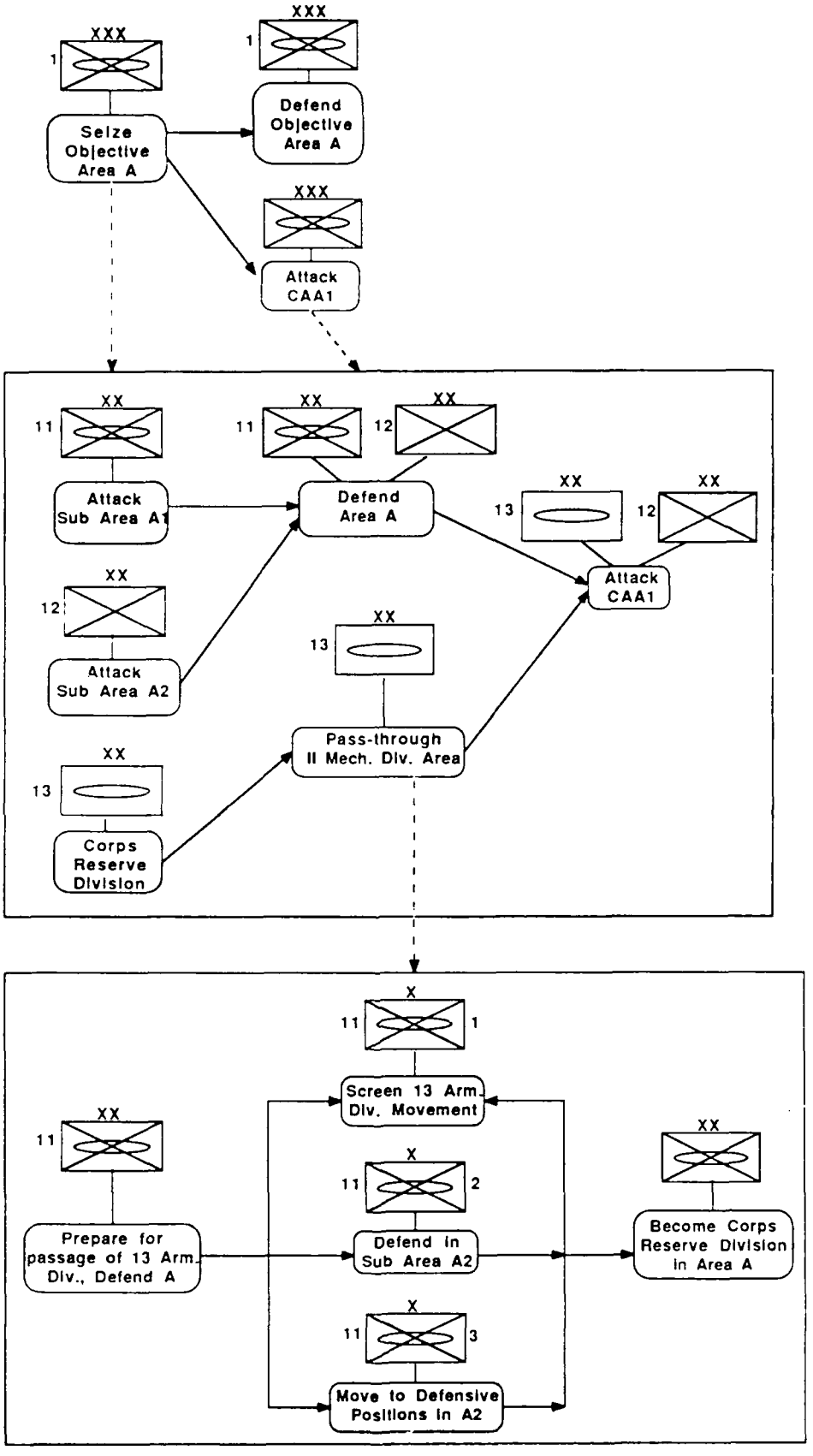


Figure 3-7: Corps Activity Represented as a Hierarchy of Activity Networks

trafficability conditions, as well as assessing potential support by supporting forces such as field artillery, or an aviation battalion.

Internally, however, the computer must have a representation that is compact and efficient to enter and retrieve data from and to reason with. This will be achieved through using a hierarchy of symbols, pointers and constraint checkers organized under the set of symbols provided in Figure 3-6.

3.6 ENTITY AND ACTIVITY FRAME REPRESENTATIONS

Large hierarchical structures lend themselves very well to representation of their elements within the structure by frames. Figure 3-8 presents two example frames: one for corps and one for a subordinate division. Entries in a slot in a frame can represent another frame, as indicated by the 13th Armor Division, being one of the subordinate units occurring in the 1st corps frame and itself a division frame. Another advantage to frame representations for this type operations monitoring problem is that there are typically only a finite number of generic-type entries that can be made for each of the frame slots. Thus, of all of the very large number of objectives that might be pursued by a corps, there are only a few generic-type objectives. In the context of a particular scenario, it is easy to specify the specific candidate objectives likely to be pursued by a corps.

Another advantage of this type representation is that when consideration is being given to filling a slot with a specific value, constraints attendant to filling that slot are quite often specific to the particular type of slot entity and can be "attached" to the slot. If constraint checking, which will be explained a little later, is used to determine the best slot filler, then it is often the case that the same constraints propagate along the pointers to the subordinate, or parent frame entities. These desired constraint propagations have been facilitated by frame reference language (FRL) constructs that have found their way into many of the supporting expert system tool environments. Figure 3-9 provides examples of a tactical operations template, or activities template. Again, we can have a library of templates, of the generic types of operations that corps, divisions, brigades are likely to perform. Each of these generic templates contain slots that can be used to deal with commander's guidance, principles of war, AirLand Battle 2000 concepts, etc. Again, in preparing for monitoring the operations to come, a default template can be selected and specific entries made for the

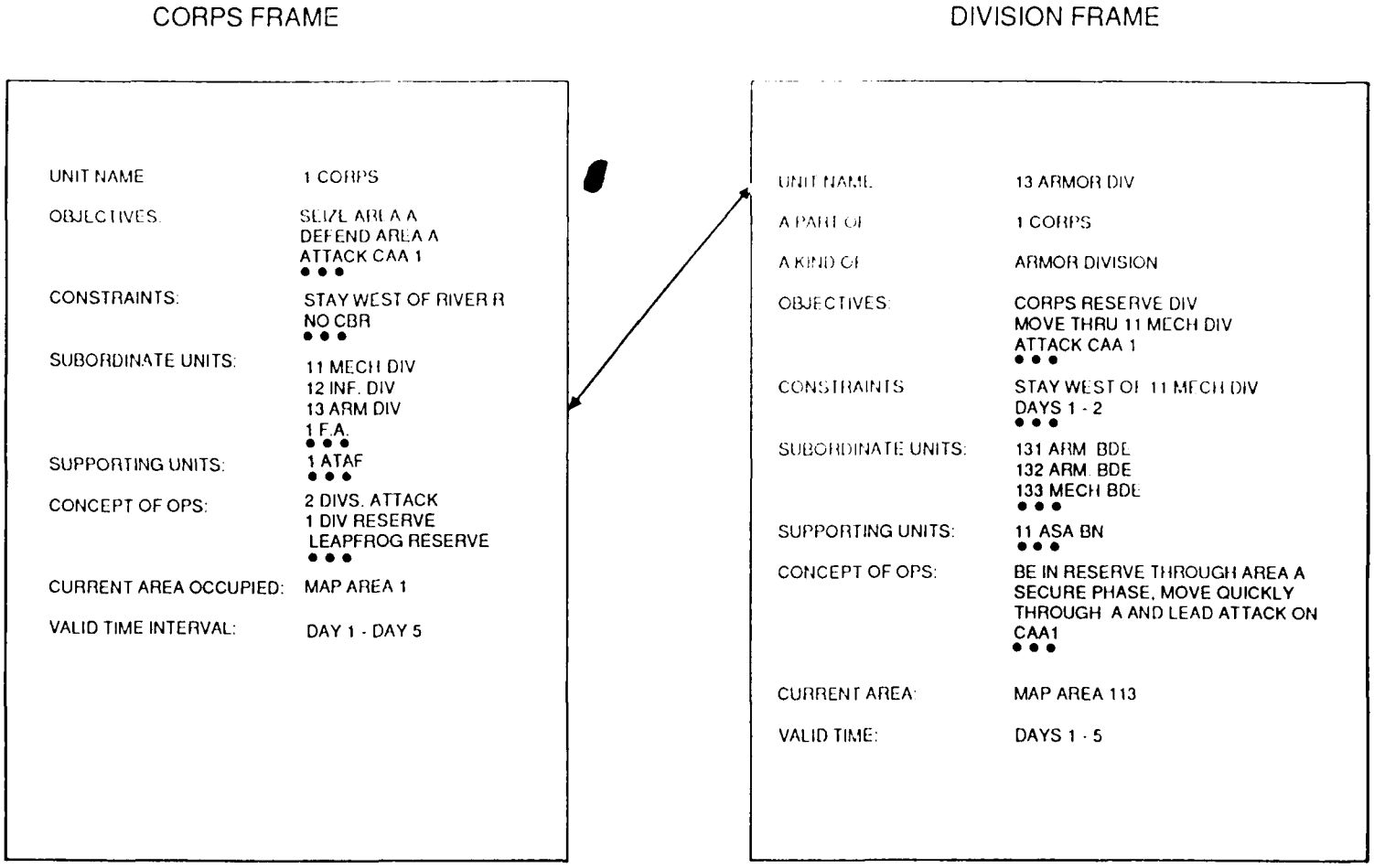


Figure 3-8: Force Unit Frames

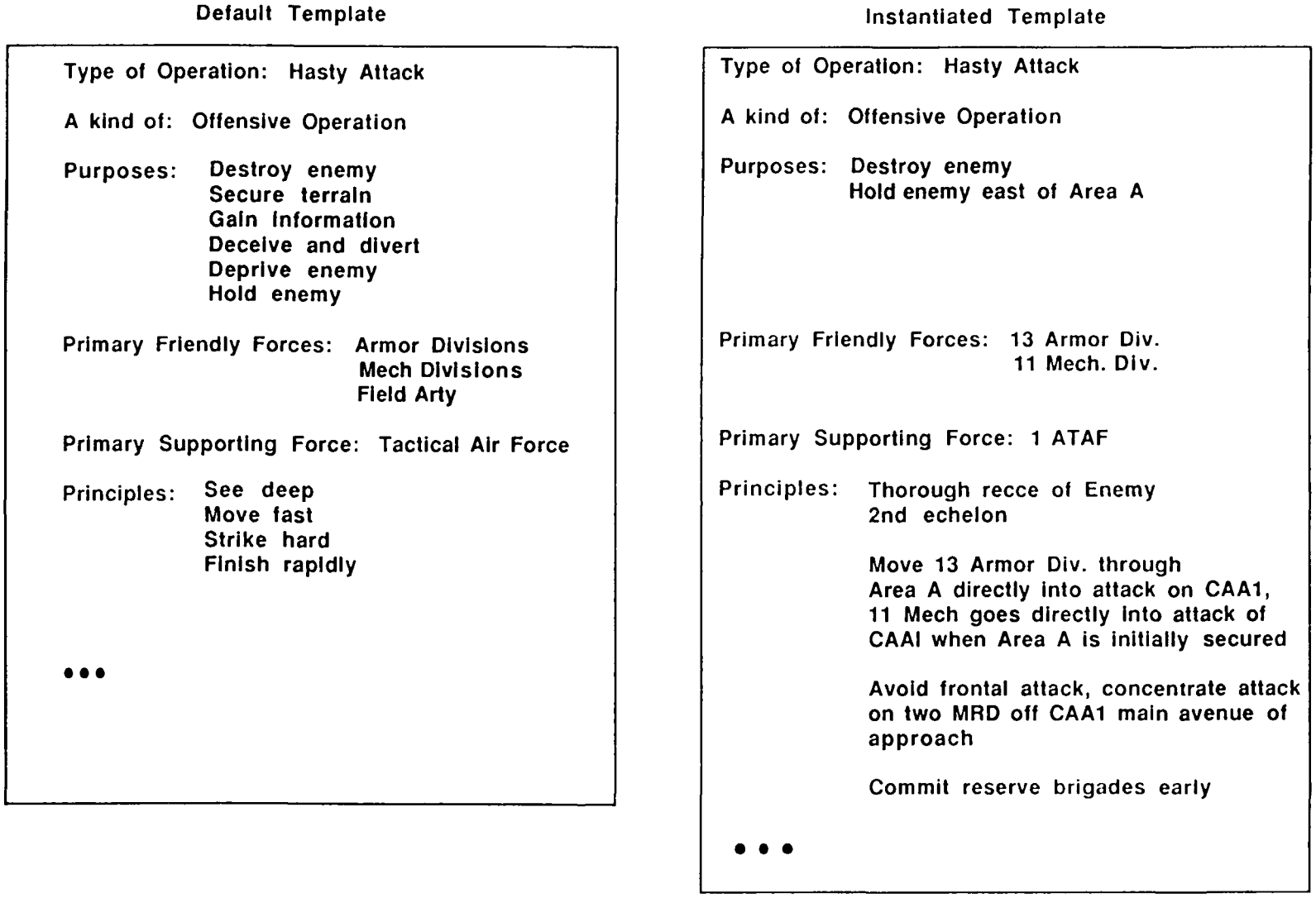


Figure 3-9: Tactical Operations Template
Examples -- Default, Instantiated

particular military situation expected and planned for. Thus, we can instantiate templates that contain a good deal of the activity information expected at each echelon level of a corps force unit. Analysis, constraint checking, and model-based reasoning can be attached to the specific slots of the activity template, thereby making OMA system process control of the procedures more efficient.

3.7 CONSTRAINT CHECKING

Constraint checking is quite often tied to, and controlled, when examining specific slot values within frame representations of an entity, or activity. In operations monitoring, when we are looking for opportunities to exploit, or situations to avoid, we must do essentially a small plan generation to find out if we have the capability to exploit opportunities, or avoid risky situations. Thus, if we get a report that an enemy regiment is in a vulnerable position because he is quite separated from his parent division, then we would like to explore the possibility of performing some type of attack against that regiment. Rather than asking the system if there is a friendly unit capable of attacking the enemy regiment within a specified item interval, a better potential plan may be generated by considering degrees of acceptability of filling the force slot for the action of attacking the enemy regiment within the specified time frame. Figure 3-10 indicates that it is possible to divide the set of potential slot fillers into a number of categories. For now, the best potential slot filler entities would be those that meet all of the constraints. If there is more than one potential slot filler that meets all the hard and soft constraints, then any of those remaining slot fillers are adequate for putting into the slot.

Hard constraints are those conditions that must be met because of physics, or hard tactics or doctrinal guidance that cannot be ignored. Soft constraints are conditions that are desirable to be met, but are not necessarily mandatory. They can be used to prioritize alternative resources, or alternative tactical actions.

3.8 TEMPORAL REASONING FOR CORPS, DIVISION AND BRIGADE SIZED ACTIONS

Most events occur over considerable lengths of time. There are a number of approaches to handling time in physical, and control-type systems, including event-based and interval-based approaches. It seems clear that an interval-based

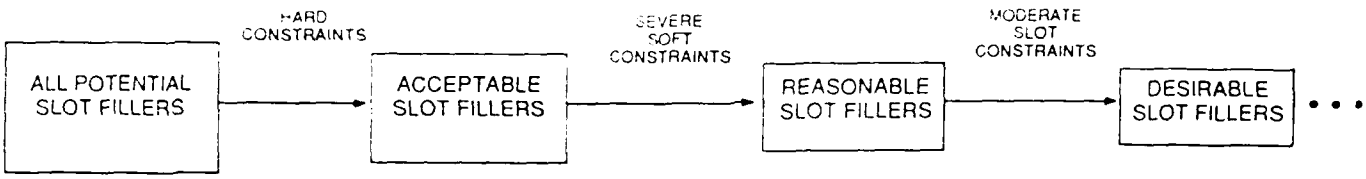


Figure 3-10: Sequence of Constraint Checking for Desirable Slot Fillers

approach should be used for operations monitoring at this level of command. Allen [1] has worked out a calculus of temporal reasoning, using intervals that is available for incorporation into our system.

3.9 OMA SYSTEM ARCHITECTURE AND CONTROL

The OMA system is a single system that could reside on a single machine, or multiple machines. As noted, it should interact with the plan generation and evaluation environment, which may, indeed, be on a separate machine. The structure and, in particular, the controller indicated in Figure 3-11 have been used on other distributed, cooperating, expert system systems. Knowledge sources and data bases may all reside a single machine, or some of them may be on remote machines and when that knowledge source is required, the controller sends, through the agenda space, a request for the knowledge source to be executed, wherever it resides. The various knowledge sources indicated can be quite simple to very complex. They can have different types of inferencing, or

model-based reasoning capabilities. They, themselves, could be a blackboard system, or a simple procedural system. They can each have their own local blackboard for posting results that are useful only to themselves. Efficiency can also be gained by permitting knowledge sources to communicate directly with other knowledge sources, rather than going through a global blackboard posting for consideration by the overall controller.

Planning data and operations monitoring data are divided into two components: static knowledge and dynamic knowledge. The static knowledge-base contains those types of information that are fixed relative to the execution of the planning session, or extended periods in the operations monitoring activity. Examples of static knowledge are given in the figure. The dynamic knowledge includes the results of inferences and calculations, and conclusions and situational information that is likely to change during operations. It also will contain partial plan hypotheses for responding to opportunities, or risk situations.

Another feature is that every time an event is posted and acted on, it goes through the agenda space and the result is sent to the history space. This history space, then, becomes a valuable source for explanations of conclusions or statements that result from using several knowledge sources.

Each knowledge source is a specialist in a particular kind of reasoning. Each knowledge source has an associated set of trigger conditions and pre-conditions that must be satisfied at the time it is executed. Keeping a truth-maintenance system table of the validity of the trigger conditions and pre-conditions can be used to speed up the control of finding the right knowledge sources to use.

This architecture permits structuring the development of the overall OMA system into relatively independent parts, which greatly aids development, but the resulting collection of knowledge sources work in a very integrated way to solve the overall problem. Thus, you can have several different people, if necessary, developing the various subsections of the OMA system; and the coordination and integration of the subsystems facilitated by the type of control and condition checking that occurs in this architecture.

4. OMA SYSTEM DEVELOPMENT PLAN

Little work has been done in the technical area of plan execution monitoring. The emphasis on building an OMA system in Phase II will be to explore the feasibility of providing an automated operations monitoring aiding capability to the G3 Operations staff; not on building a system that is an engineering prototype of a system for near term procurement. (However, we have found that intelligence personnel in operational commands have taken and directly used other "research feasibility" assistants that we have designed and built). The development plan features a research system build, evolution, and demonstration of evolving design features and capabilities; rather than the more traditional and expensive sequence of efforts that produce such things as a Systems Requirements Review, Preliminary Design Review, Final Design Document, System Build, Test and Evaluation, Acceptance Testing Documentation, Training, and so on.

4.1 EQUIPMENT SELECTION

Two major considerations should be assessed at the beginning of Phase II: what available machine best supports rapid development of feasibility models, and what other machines are in the Army command and control environment in which the OMA system would later be embedded.

The candidate machines at ADS are Symbolics, SUN-III and DEC-VAX type machines. There is a preference for Symbolics or SUN-III over VAX because of the color monitor, graphics capabilities, user interface and software development support environments. SOPE and other expert system building tools such as KEE are available for both of these machines. ADS currently has more terrain analysis software for Symbolics. The development of planning technologies at ADS is occurring on both types of machines.

If the Army decides the OMA system is likely to interact closely with the ADDCOMPE community, then SUN-III may be preferred. Other Army planning systems use other machines; in particular, the ALBMS program may use Symbolics or SUN-III's, yet to be decided.

The selection between Symbolics and SUN-III should be deferred until the start of Phase II. Either machine would be acceptable to ADS. The decision could be made immediately and would not be a block to beginning software development since ADS' computer center can provide either type to the project.

4.2 U.S. ARMY ORGANIZATION SELECTION

A particular Army organization will be selected and designated by the Government OMA program manager to interact with the ADS OMA team to provide or indicate development scenarios of interest, and domain expertise for development and review. CAC, and CGSC at Ft. Leavenworth, 82 Airborne Corps, and the 9th Infantry Division at Ft. Lewis are possibilities. Considerations such as interactions with other programs such as Battlefield Commander's Assistant, ADDCOMPE and ALBMS are relevant. ADS participation in BCA and ADDCOMPE are focussed at too low of an echelon to be a good fit for OMA Phase II, but do provide access to Army planning expertise. If ADS and team are selected for ALBMS then a great deal of knowledge engineering interaction with Army experts, and scenario development and analysis will occur and be of value to OMA. However, management care would be exercised to focus on operations monitoring technology development for whichever Army unit the government OMA program manager prefers to focus upon. This preference should be made within one month of contract award, preferably at contract award.

4.3 OMA SYSTEM DEVELOPMENT

The Phase II OMA system development will consist of:

1) Refining the design presented herein by:

- Using an agreed upon scenario to guide the choice of a terrain data base and the selection and more precise specification of types of missions, activities, types of constraints, unit templates, plan and orders forms, situation appraisal formats and "working plan" formats.

- Conducting intense knowledge engineering sessions with our

project Army expert and with Army operations experts in the selected Army organization.

- Developing a small set of default templates for force units and activity networks; and default procedural networks for specific activities (e.g., river crossing).

- Building small "bare-bones" knowledge bases and knowledge sources (KSs).

- Integrating the KSs.

- Experimenting with the above embryo system components in the computer.

- Refining technical approaches for the issues raised in this Phase I study (e.g., the ideas discussed in Sections 2.1.4 and 3.4).

- Producing a "bare-bones" prototype OMA system.

2) Reviewing the prototype design.

3) Scoping and focusing the development effort by specifying a sequence of scenario situations and types of missions and activities for increasingly more in-depth development.

4) Evolutionary development of the OMA system from the "bare-bones" prototype by:

- Conducting repeated knowledge engineering activities:

  through frequent interaction with the OMA team Army operations expert.

  — In review and guidance sessions with active duty Army operations experts.

- Encoding the knowledge.

- Running the system on scenario situations.

- Refining the knowledge base constructs and parameter values.

- Iterating on these steps.

5) Demonstrating the OMA system capability by:

- Using a scenario that is similar to, but different from the development scenario.

- Using non-team operations experts to make the decisions, but aided by OMA team members to facilitate the mechanics of using the system.

6) Identifying specific Army environment(s) in which to "migrate" the OMA system after Phase II.

Knowledge-based systems, by their very nature, are never "complete." Like human experts, they can always be both extended to incorporate a larger breadth of capability and refined to use a deeper set of knowledge. The OMA system as designed permits continuing growth in both of these dimensions, during and after the Phase II development effort. However, there is a minimum amount of development effort below which very little value could be expected to accrue. It is estimated that at least a two-man level of effort for 18 months would be required to develop and demonstrate a sufficiently capable OMA system to be able to qualitatively evaluate its potential for complete development and use in the field.

The effort would require personnel experts in AI planning technology, Army operations, and expert system "hacking." Personnel with high levels of expertise will be employed because of the use of relatively "untried" AI technology.

Figure 4-1 presents a scheduled program plan for the above efforts.

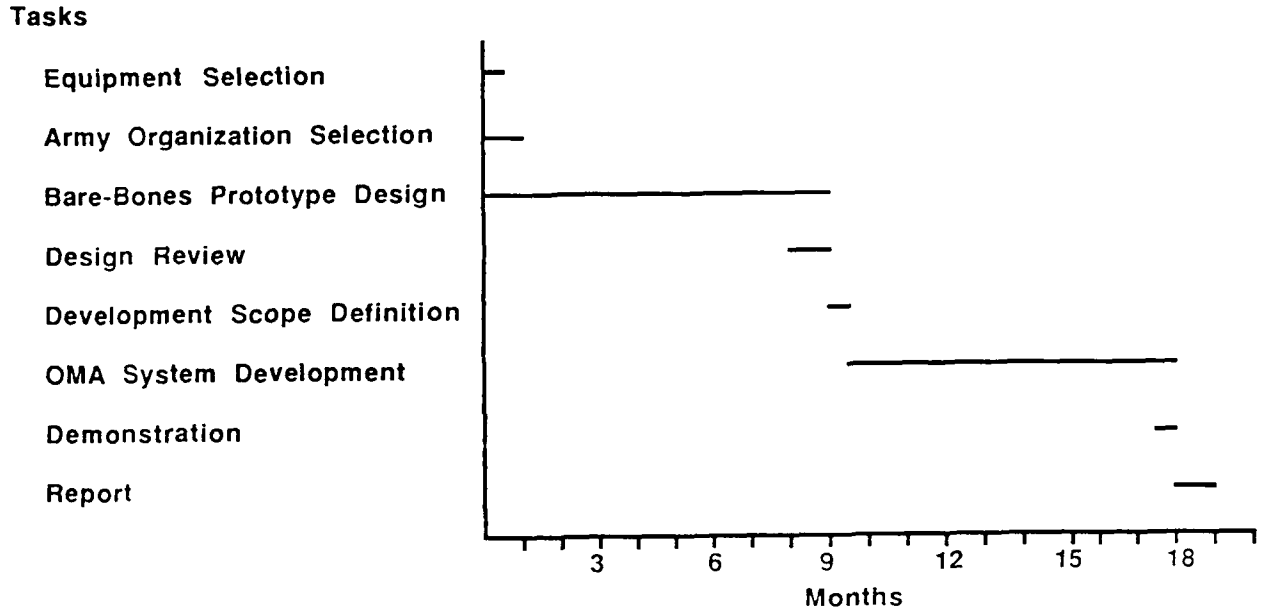


Figure 4-1: OMA System Development Plan

REFERENCES

1) FM 101-5, "Staff Organization," Department of the Army, Brachman, R.J., Levesque, H.J., editors.

2) "Readings in Knowledge Representation," Morgan Kaufmann Publishers, Inc., Los Altos, CA 94022, 1985.

3) Schoomaker, F.B., Kenney, J.L., "The Airland Battle Management Program Corps Operations and Scenario Description," BDM, October 31, 1985.

4) Brachman, R.J., Levesque, H.J., "Readings in Knowledge Representation," Morgan Kaufmann Publishers, Inc., Los Altos, CA 1985.

5) Adams, T.L., Orr, G.L., Tollander, C.J., "An AI Approach to Coordinated Fault Diagnosis, Control and Planning for the Space Station Electrical Power System," in the Proceedings of the AIAA Space Systems Technology Conference, June 9-12, 1986.

6) Erman, L.D., Lark, J.S., Hayes-Roth, F., "Engineering Intelligent Systems: Progress Report on ABE," paper in Blackboard Workshop at Carnegie-Mellon University, June 12-13, 1986.

7) Arbel, A., Tong, R., Payne, J.R., "Options Generation Techniques for Command and Control," Advanced Decision Systems, September, 1982.

8) FM 100-5, "Operations."

9) Cation, M.K., "SOPE: A Systems Oriented Programming Environment," Proceedings for the Second AI & Advanced Computer Technology Conference, Long Beach, CA, April 29, 1986.

10) Cation, M.K., "SOPE Reference Manual," Advanced Decision Systems, 1985.

[1] Allen J., "Towards a General Theory of Action and Time," Artificial Intelligence 23(2):123-154, 1984.

APPENDIX A. SCENARIO FOR A CORPS COMBAT SITUATION

A U.S. Army corps is deployed in a friendly foreign country overseas as part of a Joint Task Force (JTF). The combat elements of the corps have deployed forward from a logistic buildup area to a threatened area where a hostile power has made an incursion across the border. Contact with hostile forces has been made, and the corps now is in a combat situation.

A buildup of ten days of supplies was in progress at the buildup area when the corps was ordered to deploy forward to halt the hostile incursion. Thus, supplies just unloaded from ships had to be dispatched forward by air, road, and rail to sustain the corps in its combat operations. The corps commander has established as a goal for the logistics function a ten-day stockage of all classes of supply in the corps area forward, as well as ten-days stockage back in the buildup area. The JTF commander has concurred in this goal.

The composition of the corps is its headquarters, one mechanized division, one airborne division, a cavalry (air combat) brigade, a field artillery brigade, and several combat support and combat service support units. In a combat situation, consumption of supplies is calculated to be 7,500 tons per day. The JTF also has an Air Force component whose requirements for supplies must be balanced with those of the corps by the JTF commander and his G4.

The tactical situation as it has developed involves the defense of an airhead and a railhead from the incursion. Figure A-1 portrays the general situation. The hostile force has been identified by intelligence as a corps consisting of a mechanized division, an infantry division, a separate tank regiment, and supporting units. It has crossed a bridge over a river, which forms the border with the friendly country, and has proceeded along an axis formed by a road and rail line. Only feeble opposition was encountered against border constabulary troops of the friendly country, and the incursion has progressed over 40 miles.

Just beyond this point, a small city is on the rail line, and a small commercial airport lies about 15 miles outside the city. These facilities have been chosen by the corps commander as those around which the defense has formed. The city is the site of the railhead and the airport that of the airhead for the movement of supplies. The airborne division has been deployed to the airport and has engaged the enemy across and astride the road and rail line. It initially gave ground grudgingly but now has established a viable defensive position, which has caused the enemy to halt its main thrust. An attempt by an infantry

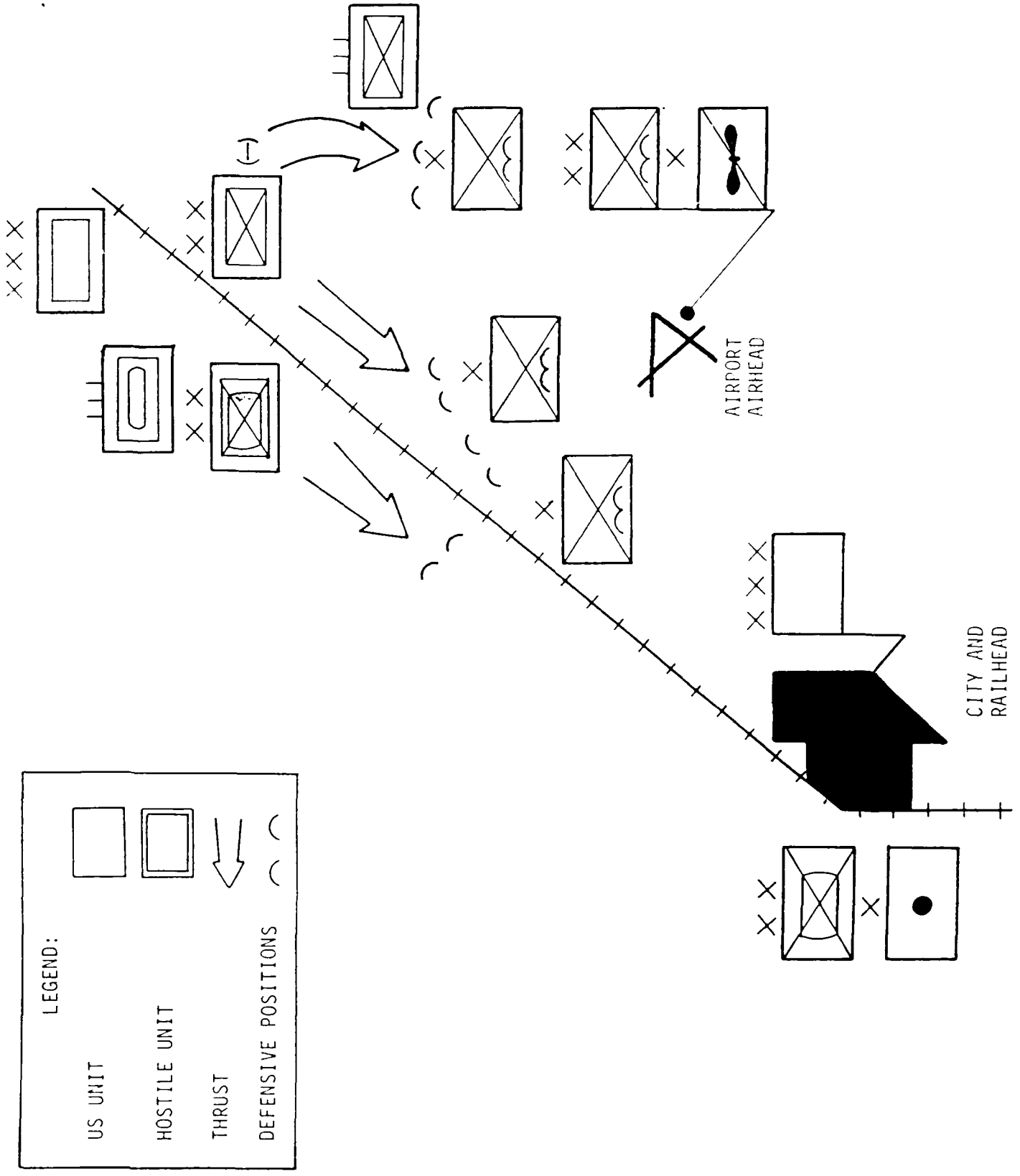


Figure A-1: Tactical Situation

regiment from the opposing force to outflank the airborne division positions has been countered by deployment of the brigade held in reserve by the airborne division commander. At this time, all three of his brigades have been committed and only one infantry battalion remains as a reserve force. Because of this, and due to the presence of tank units in the opposing force, which the airborne division is ill-suited to match, the corps commander attaches his cavalry (air combat) brigade to the airborne division. This brigade is also deployed at the airport; its attack helicopters provide the means of countering the enemy tanks.

Supplies needed for the airborne division and cavalry (air combat) brigade far exceed what can be provided by tactical airlift; the corps commander must provide enough supplies over land so that the defense being conducted will continue to hold the opposing force. However, this requirement must compete with the supply buildup which will allow the means necessary for the corps commander to reverse the tactical situation.

The "heaviest" part of the corps, and that part which uses the bulk of corps supplies, i.e., the mechanized division and the field artillery brigade, has not yet been committed to the battle. These units are in assembly areas near the railhead at the city near the airport. The corps commander is waiting for two events: the next move by the commander of the opposing force and the buildup of sufficient supplies to conduct a counterattack.

The rail and road facilities of the host country provide the main means for building up and sustaining the military force of the JTF. The corps is deployed near the border on the opposite side of the country from the port where the ships from CONUS are unloaded. Thus, its line of communication is much longer than that which extends from the port to various airfields at which are deployed the squadrons of the air force component of the JTF. Since the army component must receive support from the air force component, the prioritization of supplies to the various units is a matter of constant attention and review by commanders and chief logistics officers.

At this juncture, the opposing force commander determines to change the tactical situation. He had halted his advance to review the situation and reorganize his forces when he found himself opposed by the airborne division defending the airhead and railhead. His attempt to turn the right flank of the airborne division has been stymied. He decides to send his tank regiment across

land, off the road and rail axis, to outflank the airport defenders from the left and isolate them from the rest of the corps, which he knows to be in and around the city. He has considered his supply situation and concluded that he can sustain the armored thrust if his own supply line remains intact. Further, he has judged that the shortness of supplies available to the U.S. corps commander will not allow use of the mechanized division to oppose his own move.

To keep the airborne division occupied during this evolution, the opposing force commander orders his divisions near the airport along the road-rail line to resume their attack.

The movement of the enemy tank regiment is detected by intelligence within an hour after it begins. The corps commander is informed, but at this time the situation has not developed to the point where the enemy's course of action is clear. What is clear is that the initial move of the enemy is in a direction that is undefended and thus, provides a clear path either to the railhead at the city or to the rear of the airborne division. Soon afterward, the corps commander is informed by the airborne division commander that his defensive positions along and astride the road-rail line are under renewed attack.

The corps commander immediately arranges a meeting with his staff and the commander and staff of the mechanized division. Various courses of action are identified and discussed; the supply situation is reviewed and its impact on the various options is a critical factor in the commander's deliberations. The enemy commander's flanking attack with his tank regiment has forced the corps commander to the point of decision.

The plan which the corps commander decides to implement involves three distinct but coordinated main actions: interdiction by air power of the enemy line of communications; containment of the enemy tank regiment by the mechanized division's organic attack helicopter unit; and, most decisively, a counterattack employing a wide sweep by the bulk of the mechanized division to strike behind the enemy forces being engaged by the airborne division. Figure A-2 illustrates the main features of this plan.

The air attack on the enemy's line of communication must be requested from the air force component. This requirement includes destruction of the bridge at the border to cut off supplies from the hostile power to its force

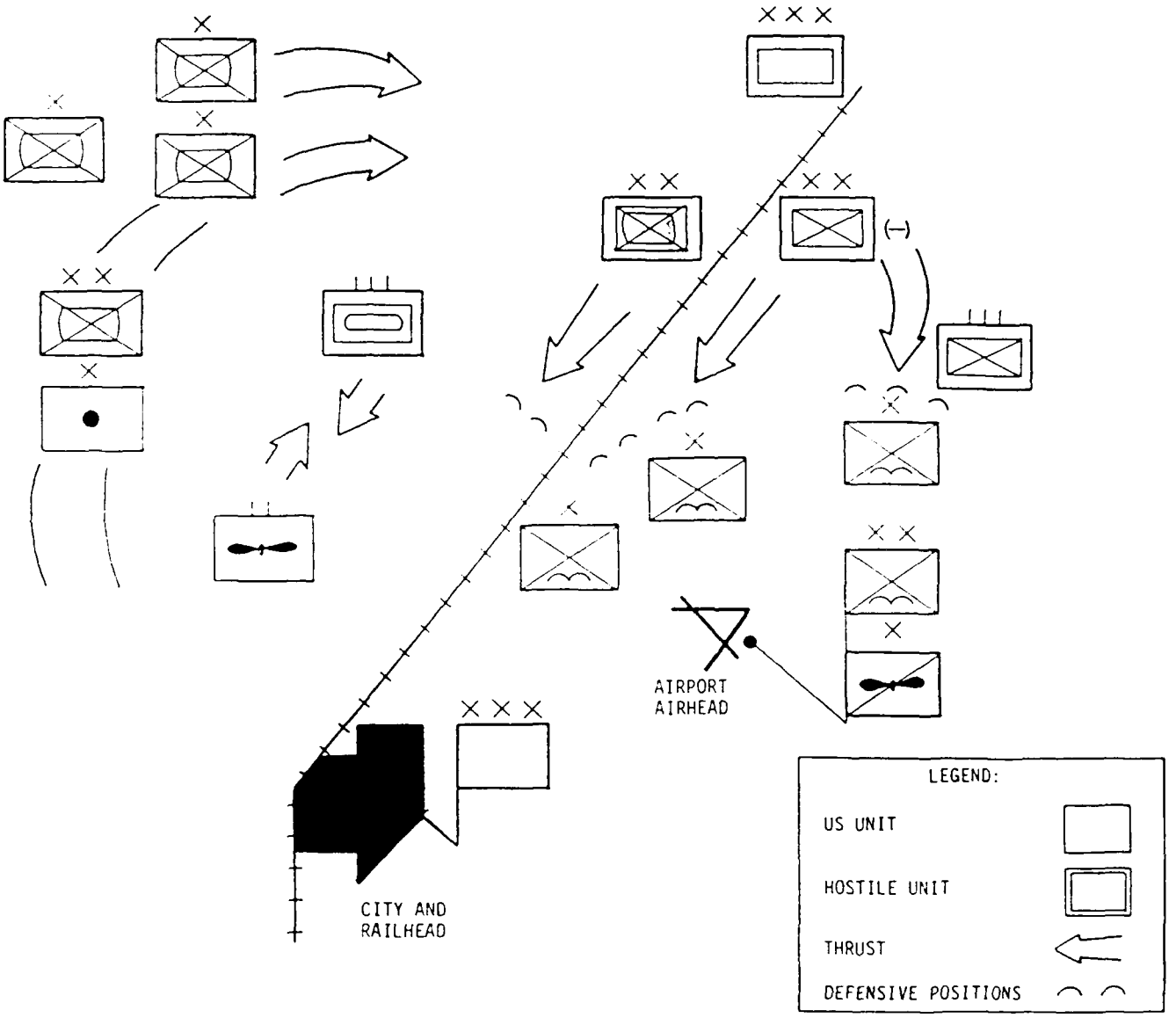


Figure A-2: Counterattack Plan

involved in the incursion. The objective is to deprive the hostile force of its means to continue its attacks. If successful, this would relieve the pressure on the airborne division and also slow or halt the thrust of the enemy tank regiment. Subsequently, the enemy would be left without the means to conduct even an effective defense.

The effects of the air interdiction will not be immediately felt, and the enemy tank regiment must be checked without delay. This necessitates the assignment given to the attack helicopter unit of the mechanized division. If successful, this would negate the danger of the airborne division being outflanked and isolated; also, it neutralizes a threat to the railhead at the city and to the right flank of the mechanized division, thus, allowing it to proceed unimpeded in its counterattack to the rear or the main enemy force.

This counterattack by the mechanized division is envisioned by the corps commander as the decisive stroke. Self-propelled artillery units of the field artillery brigade will be attached to the division. It will drain most of the supplies presently built up, but the corps commander accepts this risk because the advantages of the operation's success are judged worthwhile. The operation's success would have the mechanized division in the rear of the enemy's divisions, already in contact with the airborne division to their front. The result would be the disruption of the hostile units as an effective fighting force. The railhead and airhead would be secure and the hostile incursion foiled.

Logistics required to carry out the plan involve ensuring sufficient supplies for the counterattack by the mechanized division and the checking of the enemy tank regiment while simultaneously providing the airborne division enough to sustain its defense.

The corps G4 has the immediate requirement to ensure fuel and ordnance are available for the attack helicopters to contain the movement of the enemy tank regiment. The remainder of the mechanized division with attached self-propelled artillery is alerted to commence its counterattack in four hours.

The options the Commander exercises must place his forces in a position of relative advantage in order to be successful. In all situations, the ability to handle information rapidly is critical if the corps is to be expected to execute decisions before the enemy can react.

APPENDIX B. AI PLANNING TECHNOLOGY FRAMEWORK

Table of Contents

discuss the lowest-level issues of implementation and efficiency (e.g., caching strategies). Rather, we examine slightly higher-level considerations, such as knowledge representation languages and search control techniques.

Throughout this section we aim for breadth, rather than depth. We provide some perspective about the field as a whole, but to learn more about any specific topic that is covered, there exist a variety of general textbooks and specific articles nilsson80, rich83, charniak85, webber81, cohen82. For the interested reader, we have provided citations that should serve as pointers into the AI planning literature. We have not, however, provided anything approaching a comprehensive bibliography; the books cited above provide more complete sections of references.

B.1.2. Planning Paradigms

There are many ways to classify planning systems. In this section, we take the view that such systems can be described by the *kinds of plans* they generate, by the *way they generate* plans, and by the *way they encode* the world for planning. We briefly consider broadly descriptive categories under each of these headings.

B.1.2.1. Kinds of Plans Generated

Linear Plans Most commonly, the output of a planning system has a linear structure, that is, it is a concatenation of operators that are meant to be applied in sequence. The plan is temporally ordered (i.e., the operators' order is completely specified) and has a "flat" structure. Even when development of the plan did not at first require a total ordering of the operators, the planner will ultimately make a commitment to a total ordering, since a linear plan is the most natural detailed description of actions to be followed by a single active agent. In fact, the methods by which a partially-ordered plan is converted into a totally-ordered plan, and the stage of the planning process at which this conversion takes place, have been important areas for research.

Non-linear Plans In contrast to totally-ordered, linear plans, some planners might return non-linear plans, i.e., partially-ordered sequences of operators. For example, a plan might specify that action a occur before action d, and that actions b and c occur after a and before d, but might not specify anything about whether b or c should be performed before the other. There is in this output inherent parallelism, which may make it suitable for execution by multiple agents, acting concurrently. However, since the sequence of actions is not fully specified, care has to be taken that the various actions do not interfere with one another when they are actually carried out in the environment.

Another way in which the output of a planner might be non-linear is if it is *hierarchical* in structure. This is the case when the final plan includes operations at

different levels of abstraction. For example, a plan to paint a room might include high-level goals that have not yet been fully expanded into low-level actions, such as "buy paint," mixed with low-level atomic actions, such as "dip the brush into the paint."

A non-linear, hierarchical plan may also be produced that includes both the final actions and their associated higher-level goals, even when these higher-level goals have been fully expanded.

Partial Plans (interleaved planning and execution) When people construct plans, they generally do not plan down to the last detail all at once. Rather, an incremental process of planning is used. Part of a plan is developed and execution is begun, with the understanding that later parts will be constructed afterwards (usually when more information becomes available). For example, a plan to travel from New York to a particular St. Louis hotel may consist first of taking a plane from one airport to another. The travel on the plane is begun before it is clear how one can get from the St. Louis airport to the hotel; it is simply assumed that this part of the plan will be developed at a later time. This sort of planning *interleaves planning and execution*. It appears to be a fundamental technique that real-world planning systems will need to use, since there is rarely enough information at first for a system to complete the entire planning task. Despite the importance of this problem, relatively little work has been done on it.

This type of planning, along with conditional plans (described below), are attempts to deal with inherent uncertainty in the planning domain.

Conditionals and Iteration The most basic plans are linear sequences of actions---operators that are strung together to move from an initial configuration of the world to a final configuration. Many problems, however, cannot reliably be solved by a simple sequence of steps. There may be uncertain conditions that need to be checked (e.g., is a door open or closed), or a sequence of steps that need to be repeated until some condition is true. These control structures of *conditionals* and *iteration* can be used to construct a more complex plan, one that is more general and capable of dealing with various real-world circumstances.

In the limit, one might be willing to think of a plan as having all the control constructs found in a computer program---though the plan, unlike the program, will need to handle uncertainty in the environment. This level of control generality is rarely found in the AI planning literature.

Generation of Abstract Plans (plan learning) Learning is an important topic of AI research, and the planning literature makes its own contribution to this field. Several planning systems have the capability of generating *abstract plans* that can then be used in subsequent plan generation---instead of having to search for a plan from scratch, a planner can use a previously discovered plan (found as the solution to a

similar problem) as an aid in finding the new plan. For example, if a planner has been told to find a plan that results in block C being on top of block B, and in block B being on top of block A, it may construct a particular sequence of operations. Now, if a similar problem is presented to the planner, but this time it involves blocks F, E, and D, the planner may be able to use its previous solution in solving the new problem.

The technique used in plan learning involves abstracting out the key features of the previously developed plan, so that it can be used in the new context. This involves, for example, turning constants that appear in the plan into variables (in our example above, turning the block names A, B, and C into variables x, y, and z, which could then later be instantiated as D, E, and F). This capability was a part of the STRIPS planner (where it was called the MACROPS facility) and part of the HACKER system. Finding useful techniques for generalizing plans and developing abstract plans remains an open research issue.

Plans for Multiple Agents There has been recent work done on the problem of planning for multiple agents. The primary purpose of distributing plans among various agents is increased efficiency (due to parallel execution of the plan). In addition, the planning domain sometimes consists naturally of a group of geographically distributed agents, or of functionally diverse agents; development of parallel plans allows for a straightforward mapping of actions to agents. Plans for multiple agents can be developed at a central location (as is usually the case in current systems), or can themselves be developed in parallel by planners working on their own.

The chief issue in developing plans for multiple agents is to ensure these agents' cooperation with one another. In particular, the actions of the agents must be *synchronized*, and the agents must avoid *inadvertent destructive interference*. In certain cases, explicit communication is used by the agents to achieve this cooperation; at other times, the cooperation is achieved by careful, detailed development of the separate plans and by implicit communication (e.g., visual contact).

Plans for Satisfying Multiple Goals Most planners are presented with a single goal description, a state of the world that they are to "make true" through application of a sequence of operators. In practice, however, it is often useful to accomplish several goals at the same time, and it would be desirable to have a planner that was capable of generating plans that made multiple goals true. Constraints on "primary goal" satisfaction can also be thought of as independent goals to be made true at the same time that the primary goal is made true (e.g., "Move the blue pyramid onto a red block without touching any green blocks" could be thought of as consisting of two goals).

One approach taken to solving this problem has been to generate a plan to satisfy one of the goals, and then modifying it to satisfy the other as well waldinger77 . Provision must be made, in developing the plan, to "protect" certain aspects of the world during plan execution (for example, one goal should not be undone while achieving the second).

The final (multiple goal) plan is usually of the same structure as a single goal plan---that is, it doesn't *look* any different. It has simply been constructed so as to engender several outcomes in the environment. Some research on multiple-goal pl.ns has been carried out in the natural language generation domain appelt82a (i.e., planning an utterance that accomplishes several goals).

B.1.2.2. Ways of Generating Plans

Script Based Planning One method of planning involves the use of plan templates: these "skeletons" provide the abstract structure of the plan that will be generated, without specifying the details of which operators will be used to fill it out. The task of the planner, then, is to decide which template is most suitable for achieving the goal, and then determining how to instantiate that template. The advantage of script-based planning is its efficiency---it provides a highly structured guide to searching through the space of plans. However, the technique is suitable only in domains where there is a predictable regularity to the kinds of plans that will be generated, and where this regularity can be exploited by essentially doing part of the plan generation ahead of time (i.e., building the scripts into the system).

Hierarchical Planning When the search for a plan takes place at several levels of abstraction, the planner is said to be doing *hierarchical planning*. For example, a planner may have the high-level concept of "painting a room," as well as the details of how this should be done (buy some paint, buy a brush, etc.). Now, in planning some goal like "redecorate the house," the planner may discover a sub-goal such as "paint a room," without immediately converting this latter goal into its constituent parts. Some of the planning may proceed at this level (e.g., ordering of actions, deciding which room to paint first), and only later will the lower-level goals and final actions be specified.

The key point in hierarchical planning is thus that the search for a plan takes place using goals and sub-goals that are at different levels of abstraction.

Non-Hierarchical Planning In contrast to hierarchical planning, *non-hierarchical planning* involves reasoning that occurs solely at a single level of abstraction. This does not mean that there aren't goals and sub-goals, but rather that the goals and sub-goals are all at the the same level of abstraction.

Consider, for example, the following blocks world situation. There are four blocks in our world (A, B, C, and D) and a table. The table is very small, and can only have three blocks touching it at one time. Initially, blocks A, B, and C are on the table, and block D is on top of block B. Now imagine that a non-hierarchical planner is given the goal of getting block B on block A. In order to do this, a sub-goal is generated of making the top of B clear. This in turn might involve the sub-goal of moving block D onto the top of block C (since C is the only appropriate free space)---a sub-goal very similar in form to the original goal. In this example, there is thus a hierarchy of goals

(because there is a goal and subordinate sub-goals), but there is not a hierarchy of levels of abstraction. It is therefore an example of non-hierarchical planning.

Obviously, the distinction one makes between hierarchical and non-hierarchical planning depends on one's classification of levels of abstraction in the planning process (i.e., the question to be answered in deciding whether a system was hierarchical or not would be "Does planning system X make use of multiple levels of abstraction?"). Although there can be some artificiality in how one delimits levels of abstraction, it is generally clear when a planning system does or does not make use of multiple levels.

Opportunistic Planning When people plan, they often do so in a manner that takes into account the opportunities for generating efficient plans that become apparent during the planning process itself. For example, if someone has a set of goals, "pick up groceries," "visit optometrist," "go to bank," they may begin planning a route through town to the grocery store, then become aware that the optometrist has an office near that store and decide to stop in on the way. The development of different parts of the plan are interleaved with one another, and the whole process is typically modeled after the blackboard architecture of the HEARSAY II system. Certain claims of psychological validity (or similarity to human planning) are also made for this method of planning.

Planning as Debugging Another paradigm for the planning process is that of *debugging*; this approach is epitomized by the program HACKER. A rough plan is developed that attempts to reach the goal state, but which, it is acknowledged, may contain errors. Specialized "critics" are then employed to correct the plan, to debug it. Each critic is a piece of code that is directed to look for prototypical "bugs" in the plan. For example, in a plan that is supposed to achieve goal M and goal N, there would be a critic to check for the invalidation of one of N's necessary preconditions by the actions that achieve M.

Multiple Agents (planning in parallel) Above, we mentioned research that is being pursued in designing plans for multiple agents. There is also research going on in the parallel design of plans, i.e., plans that are fabricated by agents working in parallel. Inevitably, these plans, once constructed, are intended to be carried out by distributed systems of agents.

There are many potential advantages to parallel planning. There is the added efficiency of the planning process: local information does not need to be transmitted to a central location (saving time and transmission costs), and parts of the plan can be developed in parallel (so that total plan generation takes less time). There is also the possibility that localized information can be more effectively brought to bear on the planning process; for example, the search space for each agent using local information may be much smaller than the search space for a centralized agent using global information. There is also a potential advantage of robustness, since plan generation

does not depend on a single agent that might fail.

One drawback to this paradigm is that, eventually, the pieces of the plan must be merged (or at least synchronized) so as to avoid destructive interference of the subparts. In general, the interactions among sub-plans can be quite subtle, and it is not always a simple matter to coordinate or patch them so that they don't interfere (though this coordination and patching has been the focus for most multi-agent planning research rosenschein82, georgeff83). Sometimes, it will be necessary to build an entirely new plan.

Another drawback is that as soon as the planning becomes distributed, one must consider issues of communication in the plannning process. For example, one agent may have local information that is critical to the planning process of another agent. To get effective planning, each agent must decide **what** information to communicate, and **whom** to communicate it to. These are difficult issues.

Of course, the communication situation is only marginally better in the centralized planning case (where one agent develops a plan for many agents to carry out). There are still difficult communication issues---a local agent cannot realistically tell the centralized planner everything about its current situation, and must instead decide **what** relevant facts to transmit. In this case, though, at least the agent need not decide with whom to communicate; there is one central agent who can use the information.

Reactive Planning The AI literature has recently adopted the term *reactive planning* to designate the related work of several planning researchers who are attempting to construct systems that react intelligently to changed circumstances. Reactive planning ties together the concepts of interleaved planning and execution, the construction of partial plans and dynamic plan expansion as execution progresses. The planning component and the execution component are closely linked, with new planning taking place when new information becomes available (e.g., updating of beliefs, updating of goals, construction of new courses of action that reflect a new reality). The key problems that this work addresses are the issues of planning in an uncertain world with incomplete information, and in a world where the planning agent itself has bounded resources (i.e., it cannot spend an indefinitely long period of time planning or replanning). Relevant work includes georgeff86, lansky86, finger86 .

Adversarial Planning Most planning research assumes that the world can be described in terms of static and dynamic objects, with associated attributes but no associated intelligence. For example, the planner's world may consist of rooms, blocks with associated colors, etc. Some recent work has considered the possibility of the planner's world becoming complicated by the presence of other intelligent agents, but usually these agents are assumed to be "benevolent" rosenschein85a , in that their goals are not in conflict with the planner's goals and, given certain constraints, they are

always willing to cooperate with the planner.

In some situations, however, the planner may need to assume that its environment contains potentially hostile agents. In this case, the planner must take into account their goals, which may be in conflict with the planner's own goals. Sophisticated "adversarial planning" is a relatively unexplored area of AI, though some work has been done on these issues genesereth86, rosenschein86, carbonell81 .

There was early research in AI that examined simple adversarial relationships, such as that between opponents in a game (e.g., checkers, chess) samuel59, nilsson71 . Search techniques were used to evaluate options, under the assumption that the actions of the two sides were interleaved, and that there was full information (though possibly of only approximate accuracy) about the relative "goodness" of different outcomes. Newer work considers situations where the adversaries will be taking effectively simultaneous actions rosenschein85a , and where full information may not be available.

B.1.3. Ways of Encoding the World for Planning

There are a wide variety of alternate methods for encoding descriptions of the world in AI planning, and one can usefully characterize systems by the approach they take to this problem. We will not review all these methods here, but will only describe two widely differing approaches that demonstrate representative potential techniques. We describe the second in more detail than the first because (being more recent) it is less well known, and is more difficult to find described in the AI literature.

Logical Deductive Planning---Beliefs, Desires, Intentions One popular paradigm for "formal" planning involves the explicit representation of the planning agent's beliefs, desires, and intentions (BDI) konolige84, cohen86 . These aspects of the automated agent, corresponding to the analogous psychological components of a human agent, are then manipulated within the computer to arrive at what is often called rational action, i.e., action that is in pursuit of reasonable goals. The notion of beliefs corresponds roughly to the *facts* that an agent has in its database (and sometimes also to the facts it can derive). Desires are related to the *goals* that an agent possesses, and intentions relate to the *plans* of an agent.

The representation of an agent as having these components leads to an intuitive and formally precise system, in which a variety of planning techniques can be embedded. Research continues on ways of formally describing some of the necessary mechanisms of rational action (e.g., how to reconcile conflicting goals).

Situated Automata An alternative to the Belief, Desire, Intention model of representing an automated agent has recently been proposed rosenschein85b . Instead of explicitly building into an agent an encoding (in internal data structures) of facts, goals, etc., the agent is seen as a *situated automaton*, which enters into various states

based on its sensory information and computational efforts. These states bear no direct syntactic relationship to the logical encoding of facts (though there is a semantic relationship).

For example, consider an agent. If it were using the BDI model (discussed above), and it knew that BlockA is red, there would be some explicit data structure representing the fact (e.g., a logical statement "RED(BlockA)"). But what if the agent is, instead, a situated automaton? In this case, if the agent knows that BlockA is red, there will be no explicit data structure in its memory such as "RED(BlockA)." This knowledge would be *implicit* in its current state. It would be reflected in any actions the agent would take in this state, or in the transitions to other internal states that the agent might undergo.

There is a consistency to this view, since even in the BDI model there isn't *really* a data structure "RED(BlockA)"---there are just 1's and 0's inside the computer (or electrons flowing through gates). The use of abstract notions like "RED(BlockA)" just provides the agent's designer with a useful conceptual level at which to consider the problems of rational action. Problems potentially arise when this designer tries to actually build a program that operates on statements like "RED(BlockA)," since then what was once a useful conceptualization of rational action may become a burdensome computational inconvenience.

But the situated automata approach still takes a formal view of the design of rational agents. In particular, logic is still used in the situated automata paradigm to describe an agent, but it is used *by the designer* to specify the characteristics of the automated agent. The logical specification is compiled down into a working model of the agent that no longer literally represents the original logic. One practical benefit may be that this compiled version of an agent will operate considerably more efficiently than the BDI models have.

B.1.4. Component Reasoning Technologies

When an automated agent is built to operate in any particular domain, there are constraints on its design imposed both by the environment and by the expectations of the designer. Depending on what the agent is supposed to be able to do, and depending on the kinds of things it must represent or reason about, there may be a variety of planning theories that ought to be incorporated into the agent.

In this section we briefly discuss some of the various theoretical aspects of planning that can be used in implemented systems. Which of these are appropriate for any particular system will depend both on the environment in which it will be operating and on the sorts of capabilities one hopes to give it.

B.1.4.1. Temporal Reasoning and Scheduling

Any agent that hopes to act in the real world must deal, one way or another, with aspects of time. Of course, even the simplest planner must construct action sequences so that the constituent actions are performed in the right temporal order. However, there are several other ways in which temporal reasoning can be involved in automated planning, including the following:

- An agent may be given goals that involve specific points of time (e.g., "be at location x at 3pm"), or that deal with ranges of time (e.g., "be at location x before 3pm but after 2pm").

- If planning is "real-time" (that is, the time it takes in deciding how to act is relevant to timely performance of the actions), then the automated agent must also factor into its planning activity the time taken doing the planning. If an agent must be somewhere in one hour, it should not take two hours developing a plan on how to get there. This issue relates to the interleaving of planning and execution, and to the problem of partial planning. An agent that suspects immediate action is required may decide to start an activity before it has fully constructed the final plan, trusting that plan expansion can continue later on.

- Synchronization and coordination of activity may need be accommodated by the planner. In its simplest form, of course, this is the problem (mentioned above) that is handled by virtually every planner---the coordination of its own activity in pursuit of goals. In more complicated environments, the actions of other agents may need to be synchronized and coordinated (this is most apparent, obviously, in a multi-agent system). We will call this kind of temporal reasoning activity "scheduling."

There have been two approaches in the planning literature to the problems of time. Some researchers (such as Allen, McDermott, and Shoham allen84, mcdermott82, shoham86) have put forward *general theories of time*, formal treatments of the problem of reasoning about time. These theories allow an agent to consider in its planning such issues as ordering and synchronization of activities. One axis along which these theories differ from one another is in how they represent time for the purposes of reasoning about it. Some theories allow reasoning about *points* of time, while others deal with *intervals* of time.

In contrast to these formal theories are the *naive theories of time* found in a great many other systems. Here, time is a simplified construct that is present in a bare form that allows the agent to perform only the most necessary planning activity---for example, the ability to string together atomic actions in a particular time ordering (e.g., STRIPS fikes71). General theories have the advantage of greater power, while naive theories provide greater efficiency. As mentioned above, the question of whether a

general or naive theory is most appropriate for a given system will be determined by a consideration both of the domain and of the designer's expectations of the system.

For a discussion of the requirements that should be met by temporal theories, see shoham85. That paper lists ten criteria by which to measure "theories of change," including the ability to deal with time intervals, continuous change, concurrent actions, and possible worlds.

B.1.4.2. Spatial Reasoning and Physical Interaction

Just as some domains require reasoning about *time*, others require reasoning about *space* and the way physical objects interact. There are a variety of different issues that fall into this general category, including:

- *Formal theories of kinematics* are particularly useful in planning movement in robotics. The motion of a robot arm, for example, must be analyzed in detail for it to accomplish its intended activity. Not only must orientations be correct, but it may be necessary to avoid obstacles during the course of movement.

- Another type of spatial reasoning is *terrain reasoning*, for example that used by an autonomous land vehicle (ALV). The geometry through which the ALV moves has its own idiosyncracies (e.g., untraversable areas, topographical features that affect speed and distance) and must be taken into account in route planning. In addition, it is often useful to reason about terrain movement at a variety of levels, planning gross movements at a high level and fine movement at a much lower level of detail.

- Just as there are naive theories of time, there are also *it naive theories of space and objects*. This so-called "naive physics" hayes85a, hayes85b is used to do general reasoning about physical reality without immersion by the planner into the details of physical laws. For example, a child is able to bounce a ball without any formal knowledge of elasticity---instead, intuitions are used that, while not strictly correct, are sufficient for the task at hand. The formalization of naive physical theories is intended to make use of this approach in automated reasoning and planning.

There is an analogy between the formal informal treatments of time, and the formal informal treatments of space. Formal theories provide generality, while informal techniques are often more efficient and sufficiently powerful for the job at hand. Deciding how formal a theory is needed (or more precisely, how *powerful* a system is needed) is a judgment to be made (once again) based on domain details and performance expectations.

B.1.4.3. Dealing with constraints

When a computer is instructed in a conventional programming language to carry out a sequence of actions, those actions are spelled out in detail---the machine is told precisely *how to do* its job. High-level languages move along the spectrum towards telling the machine *what to do*, without specifying precisely how to do it.

Planning research in AI can be seen as investigating the "what" end of this spectrum: the machine can be given a high-level description of what the user wants done, and the machine fills in the operational details. Planning in this sense bears a close relationship to work in automatic programming---but with extra considerations thrown in, such as uncertainty about the environment.

Sometimes, it is useful not only to tell the machine what it *should* do, but also what it should *n t* do. These instructions about what not to do are the **negative constraints** under which the planner must operate in constructing its course of action (and, more importantly, under which the autonomous agent must operate in the real world). When a planner is said to "handle constraints," these instructions about what the machine should not do are stated explicitly.

Of course, there may also be **positive constraints**, such as "Use resources X and Y in the plan." Again, "dealing with constraints" implies that they are given to the planner explicitly.

Constraints can be either **hard** or **soft**. Hard constraints must absolutely be adhered to by the planner, while soft constraints can be thought of as *guidelines* that will influence the form of the solution, but may be violated if necessary.

There are several other attributes that characterize the constraints that may be given to a planner:

- One important category of constraints involves **resource management**. The plan that is to be constructed must satisfy certain criteria in its use of resources. The limitations on the use of resources may exist both at the object level (in the planning domain), and at the meta-level (the resources used in planning).

  For example, a planner may may be told that it must reach a certain geographical location, but that it must do so within a set time limit. The planner is faced with a complicated time constraint that affects both which generated plans will be satisfactory (object level), and how much time it can spend on planning (meta-level).

  This example of a constraint is particularly difficult to satisfy, and few current planning systems have the combination of self-awareness and

flexibility to deal with it properly (though some have begun to address the problem dean85, dean86). A simpler resource management constraint might be, for example, to construct a plan for typsesetting and printing files that costs less than some specified amount (given certain computer-time and printing costs).

- Another type of constraint in the planning process is the problem of **illegal states**. A planner might be told that it should accomplish some task, but that at no time during the execution of the plan should the executor be in some particular specified state (e.g., "Go to room A without passing through room B").

Constraint propagation is an operation over the planner's constraints. In planning, a constraint that arises in one section of a plan may limit choices available in other sections, and discovering these new constraints can substantially cut down on the number of possible solutions (this, of course, can be very helpful in the generation of plans). The "flow" of constraints from one part of a plan to another (i.e., the formation of new constraints through the interactions of old ones) is what is meant by constraint propagation. Constraint propagation is an important and frequently used technique both in planning and in other areas of artificial intelligence, such as vision.

Consider, for example, a shipping problem. You are trying to plan the movement of boxes across the country. You have several methods of transport (car, plane, etc.), but there are a few explicit constraints: you want to move the boxes within a certain time, and below some cost. The constraints interact, because the speed and cost of transportation both rule out certain alternatives. Each can help limit the choices available, and help make an appropriate final plan easier to find.

Another approach to dealing with constraints involves the technique of **Least Commitment Planning**. Using this technique, the planner considers whether committing itself to any particular choice of an action might interfere with other parts of the plan. If there is potential interference, the planner defers the choice of a particular action until a later stage of plan development, and goes to work on a different section of the plan. The point of this technique is to avoid backtracking, that is, to avoid making a choice in the building of the plan that will later have to be withdrawn (e.g., if it leads to a "dead end").

Eventually, the least commitment planner makes choices locally that imply certain constraints on other parts of the plan. These constraints are "posted" (i.e., announced) so that other parts of the plan can be assembled taking them into account. Of course, least commitment planning can seldom be the sole method of assembling a plan, and other planning techniques are often combined with it.

B.1.4.4. Reasoning About Assumptions

One useful attribute for a planner would be for it to be able to reason about its assumptions. This type of reasoning could take several forms.

- In constructing a plan, certain assumptions are built in to the final sequence of actions. Most typically, each action is assumed to generate a set of predictable outcomes in the world, and these outcomes are often preconditions for the successful attainment of the goal state (i.e., each action's outcomes are preconditions of subsequent actions or are specified as part of the desired goal). For example, doing a PUT of block A on block B is assumed to leave block A on block B.

  Some planners explicitly check each action's predicted outcomes, to make sure that the action was successful. While this exhaustive checking improves the chances that the plan will be carried out successfully, it is a crude method, since it doesn't intelligently identify a subset of things that probably need to be checked, avoiding things that probably *don't* need to be checked.

- An intelligent planner might devote resources to checking assumptions before plan construction. For example, there may be outdated assumptions about aspects of the world that a reasoning system would choose to check, because it knows that some elements in the environment are highly dynamic, or because an incorrect assumption could have grave consequences. This sort of reasoning is not implemented in current planning systems.

B.1.4.5. Dealing With Risk

Few planners are currently capable of reasoning about risks when they weigh alternatives. For example, a planner might know certain probabilistic information about actions that it can take---there may be certain risks incurred when one plan is chosen over another (e.g., an autonomous land vehicle may be destroyed by the enemy if it chooses the wrong route). One approach to this problem is to incorporate into a planner decision theoretic techniques for evaluating alternative courses of action. The planner would still have to construct various plans, but decision theory would give it a way of considering the relative utility of pursuing different options. This amalgamation of AI and decision theory has begun in a small way, but much work remains to be done. Dealing with risk, of course, also involves dealing with inherent uncertainty, as discussed below.

B.1.4.6. Reasoning Under Uncertainty

Bayesian Reasoning

The classical methodology for combining quantified evidence is Bayes' Rule. Bayesian reasoning is characterized by quantitative degrees of support, a prior enumeration of tenable hypotheses, a prior model of the statistical support for hypotheses independent of any set of observations about them, and an underlying conceptual model based on the principle of a "fair bet." The fair bet is the fraction of the resources an unbiased observer would bet on each side of an hypothesis. All resources must be gambled on one side or the other; there is no "not sure" alternative. These degrees of support are assessed from several independent sources and then combined by multiplying them together.

In practice, the Bayesian beliefs are combined through a specific formula (Bayes' Rule) which performs conditioning of probabilities: current data are interpreted with respect to knowledge of the observation independent prior support for the given hypotheses. As a result, these "priors" have a tendency to dominate the outcome of the Bayesian belief combination process. A practical attempt to address the probabilistic degree-of-belief concept in expert systems is given in the work of Duda et. al. (1979), in which is developed a subjective Bayesian technique for representing uncertainty. This technique uses the concentional statement of Bayes' Theorem, but supposes that the probabilities are subjectively determined. Modifications are introduced to deal with the problems of interdependency that arise when dealing with a network of rules, and were implemented in PROSPECTOR.

A related practical effort to represent degree of belief is given in the work of Shortliffe and Buchanan (1975). In this model of reasoning, degrees of belief represented as "certainty factors", rather than probabilities. These are functions of two other measures, namely the "measure of belief" of an hypothesis given the evidence and the "measure of disbelief" given the evidence. The MYCIN system is built upon these concepts.

The strengths and weaknesses of Bayesian systems are well known. One of their principal weaknesses is the number and depth of the assumptions one must make in order to be able to apply a Bayesian analysis. What is the innate prior likelihood of all hypotheses we might want to consider? How do we arrive at such a set of prior hypotheses? What is the epistemological motivation for asserting that beliefs are well-represented as bounded real numbers? How much can we depend upon results of a Bayesian computation if, as is the frequently the case, the priors were chosen by statistical sampling or by fabrication instead of through knowledge of the true distribution?

Dempster-Shafer

A viable means for combining evidence is the Dempster-Shafer uncertainty calculus (Shafer, 1976), which subsumes both the Boolean and Bayesian techniques as special cases and additionally provides an explicit representation of the system's ignorance as

well as its knowledge. The major modification to the Bayesian techniques by the Dempster-Shafer is that whereas the former permit belief to be associated with the individually identified possibilities about the real world, the latter allows belief to be attributed to sets of possibilities, without requiring us to further distribute our beliefs to the members of that set.

Dempster's Rule (Dempster, 1967) is used to combine the degrees of support from different sources. It is equivalent to Bayes' Rule but incorporates a set intersection formalism to ensure that we only combine support where there is some agreement between the two sources of evidence. An additional normilization step is included to ensure that the degrees of belief fall within [0, 1], as in the Bayesian model; this normilization is achieved by calculation of a degree of conflict between the two evidence sources, which is itself a useful measure during analysis of the performance of a Shaferian system.

A significant problem with this technique is its ability to get confused when confronted with information outside of the "frame of discernment" (the initial set of possible hypotheses). This problem occurs frequently in the real world in the form of unpredicted events and, especially, sensor failure. Another major problem is the complexity of the technique. If the frame of discernment consists of 10 members, then there are 1024 susbsets to consider! There has been some work in circumventing this problem (Gordon & Shortliffe, 1985), but they tend to restrict the capabilities of the technique beyond its usefulness.

Fuzzy Set Theory In an attempt to break away from the traditional models of uncertainty, Zadeh introduced the concept of a Fuzzy Set (Zadeh, 1965). A fuzzy set is characterized by a membership (characteristic) function which assigns to any object a grade of membership of the object in the set. For regular sets, the charactersitic function would return either 0 (not in) or 1 (in). Set-theoretic operations are defined for these sets in an intuitive fashion in order to provide a formal structure for studying imprecisely quantified degrees of membership.

The fuzzy set approach deals with partial belief by introducing the concept of fuzzy qualifiers (e.g. likely, possibly, etc.) and contains some models for the interaction between vagueness and partial belief. This theory does not deal directly with the problem of combination of evidence but does provide some alternative interpretations of work such as Shafer's.

There are two major problems with the fuzzy set approach. One is how to consistently define the characteristic functions (the functions may differ between analysts). The second problem is choosing a calculus to implement the technique; most any norm conorm pair of functions will theoretically model conjunctions and disjunctions. Although there has been success in modelling the characteristic functions consistently, the operations on these functions are sensitive to both context and individual cognitive

preferences.

Truth Maintenance Systems

One of the early and predominant methods for handling non-monotonic reasoning is the truth maintenance system (Doyle ref.). This type of system records and maintains proofs by connecting assertions, rules, hypotheses or any other beliefs (statements) with justifications. There may be more than one justification for an assertion, representing multiple proofs of that assertion. A node is believed if there is at least one valid justification. When a proposition is asserted the justifications of nodes are checked, and those nodes which now have valid justifications are believed. The truth maintenance step consists of scanning newly justified nodes and the recorded justifications to find any other nodes which may now include a proof.

There are two types of justifications. The first type maintains two lists of nodes, the IN nodes and the OUT nodes. An IN node is one which is believed, an OUT node is one which does not have a valid justification (not necessarily false!). The justification for a node is valid if all the nodes on its IN list are in and all the nodes on the OUT list are out.

The second type of justification is a rule-like structure with a consequent and two lists, an IN list of hypotheses and an OUT list of hypotheses (usually empty). This type of justification is valid when each node of the IN list is in and each node on the OUT list is out.

A TMS begins by assigning default values to nodes where appropriate. Reasoning then proceeds via the accrual of evidence and the satisfying of justifications. If at some point a contradiction is reached, the dependency-directed back- tracking function is evoked. This function works its way back through the chain of reasoning, collecting the assumptions which support the contradiction. One of the assumptions is retracted but the set is saved to prevent the re- occurrence of the same contradiction. This retraction causes a node which was out to become in and thus begins another chain of reasoning.

Although dependency-directed backtracking is more efficient than generic backtracking, it still is a rather expensive process. Also, this type of problem solving only pursues one avenue of reasoning at a time thus prohibiting the comparison of competing node values. Another drawback is the rather shallow meaning given to justifications. It may be deemed necessary to justify a decision on more than whether certain propositions are in or out; the arguements themselves may prove useful. For this reason it is also not possible to assign any "believability factor" to the justifications. These considerations (except for the last) are addressed by deKleer in his "assumption-based" theory. The last issue is addressed in the probabilistic approach to "possible worlds."

Assumption Based Truth Maintenance Systems

The assumption-based truth maintenance system addresses the problems found in the standard TMS. Conceptually, a radical difference is the ATMS's ability to maintain (and compare) multiple possible solutions simultaneously. This is done by allowing the system to maintain contradictory information as long as that information is not used to derive new information. A TMS requires that the current set of IN nodes be consistent (non-contradictory). For example, if x -> a, y -> b, but X and y is a contradiction, then a TMS would not derive a or b (at least not from a node with x and y on its IN list). An ATMS, however, would derive both a and b since they come from x and y independently; it would not derive anythoing which required both x and y. This ability to reason about elements of a contradictory statement has proved to be a common need in qualitative reasoning.

The basic unit of an ATMS is the assumption. There are no justifications given in the system for an assumption; they are given by the user and thus not derived information. An assumption may be used to justify multiple nodes and one node may use multiple assumptions. When a contradiction is found and needs to be rectified, the belief in an assumption can be reversed. A justification in an ATMS describes how a node is derived from other nodes (including assumptions).

By manipulating all the possible solutions simultaneously, an ATMS avoids the backtracking problem found in the traditional TMS. Since inconsistent data may exist in a "context," it is not necessary to do all the INing and OUTing that is done by the TMS. Finally, the justifications in the ATMS are more robust than those of the TMS. Thus, with a reasonable implementation, the ATMS can generally out-perform a TMS see papers in vol. 28 (1986) of AI Journal .

B.1.4.7. Reasoning by Analogy

The ability to reason by analogy is considered by many to be an important characteristic of human intelligence, and it would certainly be useful to have automated systems that could take advantage of this type of reasoning. For example, a planner might recognize key characteristics of a situation and, by analogy, realize that it is very much like a previously encountered situation. Aspects of the two situations could then be matched, and a new plan generated along the lines of the previously generated plan.

This type of reasoning does not exist in current planners, except in the most rudimentary forms. For example, the STRIPS planner was able to generate plan skeletons from specific plans; these skeletons could later be used to generate new plans. This, however, is not really analogy, since it does not directly reason about a specific instance using another specific instance; rather, it reasons about a specific instance using a general rule (though the rule itself came from a specific instance). The study of analogy is an important topic in the field of machine learning; it has largely been

bypassed in AI planning.

B.1.4.8. Dealing with conflicts

There are a various kinds of conflict that can confront a planning program. First, there may be *inherent* conflicts among goals, or between goals and constraints: it may be impossible for a system to satisfy all the goals it has been given, or for it to satisfy those goals without violating explicit constraints that have been imposed on the solution. In this case, a flexible planner would judge, based on the situation, whether to notify the user of failure or construct a plan that makes compromises.

Often, however, the conflict with which a planner must deal is not inherent, but is the result of avoidable destructive interference among sub-parts of a plan. The system, for example, might be given a conjunctive goal ("Accomplish A and B and $\ldots$"), and find that its sub-plan to accomplish A conflicts with its sub-plan to accomplish B.

Various planners have varying abilities to deal with these avoidable conflicts among goals (or, more precisely, with conflicts among the methods of achieving each of those goals). We can identify two major approaches to this conflict resolution, depending on *when* in the plan generation process conflicts are resolved.

The Critique Approach When confronted with conjunctive goals, some planners solve each conjunct in isolation. At that point, some planners consider themselves done, and do not check their subplans for harmful interactions. Others, however, do perform this check. The most important paradigm for this check is the **critics** approach, implemented in the HACKER system sussman75 . The idea is that, once the subplans have been formed, specialized programs are called that embody knowledge of destructive interactions, and that check the subplans for such problems.

One well-known example of a "critic" was the one in HACKER that checked whether a precondition for one goal "clobbered" a brother goal. For instance, one goal might require as a precondition that block B is clear, but another goal might be to stack block A on block B; the precondition of one goal interferes with another goal. The solution in the HACKER system was to reorder the sub-plans when possible. This technique sometimes works, but is by no means a total solution to this interaction problem or others.

The notion of "critics" was an influential idea in planning, being incorporated into the hierarchical NOAH system as well sacerdoti75 . Its encoding of knowledge exemplified the procedural point of view in the procedural declarative controversy of the mid-1970's.

The Constructionist Approach Other planners construct their overall plan with explicit consideration of potential interactions. An example of this type of approach can

be found in [waldinger77]. Once a sub-plan for achieving one goal is formulated, it is marked as "protected," so that when the planner develops a sub-plan for a second goal, it will know not to undo the first.

One technique for insuring this non-interference is called **goal regression**. It involves "passing goals back over plan steps," in effect, adjusting the order of actions to avoid interference. In this approach, however, in contrast to the critics approach, there is no committing to a plan and then backing up; goal regression can also solve planning problems that will cause difficulties for any planner that simply reorders goals (since it works out a finer-grained adjustment of actions). For more information, see waldinger77 .

Multiple-Agent Conflicts The study of sub-plan interaction was an early, popular area of AI planning; more recently, the problem of conflict among multiple agents has received some attention in the planning literature. Although there are certain similarities between the two cases, multiple agent conflicts can introduce two new considerations into a planner.

First, conflict among multiple agents may be impossible to resolve at the planning stage, if the planning has proceeded in parallel (i.e., at different agents) and pre-execution plan comparison is difficult. In this case, conflict may need to be resolved during execution. Second, when multiple agents are involved there may be true conflicts of interest; this generally does not occur when the planning has taken place at a central location (even when conflicting goals are present, there can be a suitable local method of resolving them, which by definition will be the "accepted" method of resolution).

B.1.4.9. Planning Communication

When groups of machines must interact, or when machines interact flexibly with humans, communication can play an important role in coordinating activity. Planning research in recent years has begun to examine how computers can intelligently plan their communication activity. This research falls into two categories: planning communication among machines, and planning communication intended for humans.

Communication Among Machines In a sense, this area of communication planning is the "purer" of the two. Since the intended recipient of the communication is a machine, the researcher could assume that the generator and receiver have a common language. He can then focus on the issues of planning abstract communications without worrying how to convert this abstract structure into another language (as is necessary when humans are the intended recipients of the communication). In fact, this assumption is regularly made.

The main thrust of this research has been to consider communication as a form of

action---the planner constructs plans that include *communication acts* as well as more conventional domain acts. Work has focused on such issues as the formal semantics of making commitments, as well as the obvious issues of converting internal goals and representations of the world into communication primitives (such as **inform** and **request**). Representative work includes cohen78, cohen79, cohen86 .

Planning Natural Language Communication Man-machine interaction has been an important topic for AI research, and a crucial aspect of this interaction is getting the computer to generate easily understood messages for the human. While such a message need not be in a natural language, such as English, it has long been felt that such a capability would dramatically increase computers' effectiveness in everyday settings. Thus, much research in AI has focused on getting computers to plan natural language utterances.

Generally, the problem of natural language generation is divided into two parts, text planning and text production. The former involves generating an abstract model of a communication based on the information that the computer wishes to convey and a model of the user (similar to the machine-machine communication planning mentioned above). The latter involves the conversion of this abstract model into a correct natural language fragment. Text planning bears a close relationship to the rest of the planning literature, using some of the same techniques (such as scripts) for this specialized task. See appelt82a, mckeown82, hovy85, wilensky83 for some representative work in this area.

B.1.4.10. Reasoning Using Logics of Knowledge and Belief

"Logics of knowledge and belief" are formal methods of representing within an automated agent the facts that it knows or believes. Important research is currently underway in this area, and in specifying ways of reasoning about this knowledge and belief. For example, researchers want to formalize how one agent could reason about another agent's beliefs, or how it could reason about its own beliefs. Roughly, these various theories can be divided into two categories, *syntactic theories* and *semantic theories*.

Syntactic Theories Syntactic theories assume that an agent only believes facts that explicitly appear in its database. Another agent can then explicitly represent those facts and reason about them more or less directly. Typically, predicates such as "K" are used to represent an agent's explicit knowledge of some fact (e.g., K(Joe,"Red(BlockB)") means that Joe knows that BlockB is red). Representative work includes haas86, konolige84 .

Semantic Theories Semantic theories assume that an agent believes not only the facts that are explicitly present in its database, but also other facts that the agent *could* derive from that base set of facts (even though it has not yet done the derivations). The

semantics of possible worlds is commonly used in this approach: we say, for example, that an agent knows fact A if A is true in every possible world that is compatible with the agent's knowledge. See, for example, moore85a, appelt82a .

While semantic theories provide an elegant method of representing knowledge and belief, they suffer from several problems. First, they are typically much less efficient to use when reasoning is automated in a computer. Second, agents in the real world never actually know all the consequences of their current beliefs, and their actions reflect this lack of total knowledge. Unfortunately, the possible worlds approach assumes that they *do* know all the consequences of their beliefs. This inability to represent limited deduction is a very serious handicap, and is one of the prime reasons for researchers' interest in developing alternative, syntactic, theories of knowledge and belief konolige85 .

B.1.4.11. Logics for Planning

Among that community of artificial intelligence researchers who have pursued a formal, "neat" approach to planning, there has been much effort expended on the issue of what logical system to use. This choice, of course, affects both the epistomological and heuristic aspects of the representation, i.e., both what can be represented and what can be deduced by the system. A few of the many choices are:

Procedural Logic A formalism for explicitly representing and reasoning about sequences of actions for achieving particular goals georgeff85 .

Probabilistic Logic A form of logic in which the truth values of sentences, instead of being simply "true" or "false," are probability values (between 0 and 1)---useful for reasoning with uncertain knowledge nilsson86 .

Modal logics These are logics that deal with the concepts of *necessity* and *possibility*---often realized through the addition of so-called modal operators and axioms to classic propositional or predicate calculus.

Multi-valued Logics As with probabilistic logics, alternative logics have been proposed that consider sentences to have more than two possible truth values, that is, more than just "true" or "false." However, multi-valued logic does not assign a probabalistic interpretation to those multiple values. For example, although many values may be permitted, any value in a continuous range is not permitted (as would be in a probabilistic logic). This is distinct from theories such as Dempster-Shafer that use *intervals*.

Nonmonotonic Logics Classical logic requires that once a fact is true, it remains true for all time; this property is called *monotonicity*. Nonmonotonic logics mcdermott80 , in contrast, allow for the possibility that when

new facts are added, old facts that were once true will no longer be true. Nonmonotonic logics and other methods of doing nonmonotonic reasoning are an important area of current AI planning research. We discuss these issues in greater detail below.

B.1.4.12. Nonmonotonic Reasoning

Specifying the semantics of nonmonotonic logics has occupied a considerable amount of energy over the past few years. The facility to do nonmonotonic sorts of reasoning has clear importance in building real-world systems (since propositions relating to the real world change their truth values dynamically over time). While ad hoc schemes trivially do such nonmonotonic reasoning, it has proven difficult to adequately specify the formal theory that underlies that activity.

There are several popular alternatives for formalizing nonmonotonic reasoning: default logics, circumscription, and autoepistemic theories.

Default Logics Often, it is useful to have a system deduce some specific fact unless it can prove that the fact is false (e.g., a system might assume that any specific bird can fly unless it is explicitly told otherwise). The facility to do this sort of reasoning within a formal logical system has been developed in *default logics* reiter79. Obviously, a default logic must be nonmonotonic, since anyone using it may jump to conclusions that can later be shown to be false (i.e., you may be able to deduce propositions that later, with more information, you will not be able to deduce). Such a logic is said to be *defeasible*, that is, its conclusions are tentative. When more information comes along, its conclusions may have to be withdrawn.

Circumscription Circumscription mccarthy80 is a formalized method of "jumping to conclusions," so that a program can conjecture that the objects that it knows have a certain property are the only ones that do, in fact, have that property. It is a form of nonmonotonic reasoning, in that the addition of new facts can invalidate previous conclusions.\footnote{It is not, however, a nonmonotonic logic---it is a form of reasoning that augments first-order logic by using the circumscription schema. The interested reader is referred to McCarthy's original paper mccarthy80.}

Suppose, to use McCarthy's example, you are given a puzzle to solve, like the Missionaries and Cannibals problem (e.g., three missionaries and three cannibals must cross a river in a two-person boat without cannibals outnumbering missionaries on either bank of the river at any time). Obviously, a solution to the problem involves finding a suitable crossing schedule that satisfies the constraints. But a person, when confronted with this puzzle, might say "Have all the missionaries and cannibals use the bridge upstream." When told that there is no such bridge, he might respond "Then have them use the helicopter," and so on (there are an infinite number of such extensions to the basic world given in the problem).

Intuitively, we know that such solutions are invalid---to solve the problem we must be willing to assume that the objects that we know about are the only *relevant* objects. In order to have a machine logically deduce such a conjecture, it is necessary to use a technique like circumscription. However, circumscription (so far) is a technique for *checking* conjectures---the user must still operationally supply the predicate to be circumscribed, which limits the practical usefulness of the technique.

Autoepistemic Reasoning *Autoepistemic reasoning* moore85b is a logic that describes how an ideal rational agent would reason about its own beliefs. One of the uses of autoepistemic reasoning is to "jump to conclusions," as one does in default reasoning.

Consider an agent that believes that Fred is a bird, and that birds can generally fly. Does the agent believe that Fred can fly? If the agent has a rule to the effect that "If x is a bird, and I can't prove that x *can't* fly, then I'll believe that x can fly," then in the absence of other information, the rational agent ought to believe that Fred can fly.

There is a subtle but important difference between this approach and other default reasoning systems. In normal default reasoning, the agent in the previous paragraph could *deduce* that Fred could fly, which may not be a valid conclusion (i.e., it may not be true in the real world that Fred can fly).

In autoepistemic reasoning, however, the agent would only reach a weaker conclusion. The agent would determine that he *believes* Fred can fly, which *is* a valid conclusion---based on his information, he believes Fred to be a bird that flies. If more information comes along, the agent may no longer believe that Fred flies, but the original conclusion, relative to the original set of beliefs, will still have been valid. With that original information, the agent did believe that Fred flies. Such a conclusion is said to be *indexical*, in that it is "indexed" to the beliefs under which it has been derived.

So autoepistemic logic is a way for an agent to reason about what he believes. Such reasoning is useful for jumping to conclusions based on default rules. For any fixed set of beliefs, autoepistemic reasoning is a valid form of inference (i.e., unlike default reasoning), but as the core beliefs change, an agent will reach new conclusions about what he believes.

B.1.4.13. Possible Worlds Formalisms

There are two uses of the term "Possible Worlds" that are currently used in the AI literature. One is derived from work done by Saul Kripke, and the other is derived from work done by David Lewis.

Kripke's Possible Worlds A popular treatment of knowledge and belief in recent years has been "possible worlds" formalisms, that consider the actual state of affairs to

be simply one of many potential realities. Although an agent may know many facts about the real world, it may be in the dark about other facts. An agent, though, is said to know some fact A if A is true in all possible worlds that are consistent with the beliefs of the agent.

Using this formalism, it is not necessary to actually check through the facts that are true in various worlds; alternative worlds are fundamental to the semantics of the formalism, but not to its operational usage.

Lewis's Possible Worlds Another use of the term "possible worlds" considers the explicit existence of these various worlds. Thus, it may be useful to actually consider the facts that are true in different worlds, compare those facts to the facts true in the current world, etc. ginsberg86a. This is similar to the "multiple worlds" approach taken by various AI programming systems, where alternative realities can be considered by the reasoning component.

B.1.5. Implementation Technologies

In this section we briefly review some of the more basic techniques that are used to implement the theories described above. We are concerned here with approaches that can be used across a variety of planning systems. This is by no means an exhaustive list, but it should provide the flavor of some methods that are available.

B.1.5.1. Backward and Forward Chaining

The objective of a search procedure is to discover a path through a problem space from an initial state to a goal state. This procedure may proceed from a set of initial conditions and work toward the goal or, as an alternative, the tactic may be to begin at the goal state and work backwards toward the initial state. These two approaches are known as forward and backward reasoning (chaining) respectively.

Forward reasoning starts with a collection of known facts (the data base) and repeatedly cycles through all the rules, applying the facts and adding new ones as they are derived, until no rule runs. The principle is that during each rule firing cycle, new facts will be instantiated, thus allowing additional rules to be fired in the next cycle, until eventually a goal state is reached or the system has exhausted its possibilities.

An operational system may contain many rules and a large data base of facts. The forward reasoning described above can run through many cycles and generate a large number of new but irrelevant facts before the desired goal is derived. The backward reasoning approach may be used in this situation to increase efficiency because it allows the focusing of effort on those rules related to the results to be proved. Backward reasoning takes advantage of the potential for bidirectional operation of production rules to determine the validity of an assertion.

Backward reasoning works in the following manner. First the fact to be inferred is designated. The data base is searched for this fact, and if it is found, the fact is true and the problem is solved. If not, then the rules with the desired fact in their consequent are placed in a list. If this list is empty, then the system with the current rule set cannot solve the problem. If all the antecedents of any of these rules exist in the data base, then that rule is run, the desired fact is placed in the data base and the problem is solved. If there are rules with the goal fact in their consequent but whose antecedents are not all contained in the data base, then these missing antecedents are classed as subgoals and the above process is repeated for these new subgoals.

It is possible to combine forward and backward reasoning into a single algorithm known as bidirectional search. One potential advantage is to reduce the total number of state spaces expanded. The justification for this is that the number of expanded states in a search tree often grows exponentially with the depth of the search. Thus if the expansions could begin simultaneously from the start and goal states and meet approximately at some medium depth, the potential combinatorial explosion of expanded nodes at the end each expansion could be avoided.

B.1.5.2. Blackboards

Another expert system framework is the blackboard architecture. This approach was developed to cope with the following computational problems.

1. Integration of multiple sources of knowledge.

2. Problems whose solution depend on heuristic methods and noisy data.

3. Computational complexity.

4. Integration of different problem solving methods.

5. Potential for organizing parallel problem solving activities.

The term blackboard refers to a central data base used by systems with this architecture to coordinate and control the operation of independent groups of rules called knowledge sources. The knowledge sources communicate by writing messages on the blackboard and reading messages from other knowledge sources. The blackboard architecture has four distinct components, entries, knowledge sources, the blackboard and a control mechanism, which are described below.

Intermediate results, called entries, are generated during the problem solving process. Entries can be elements of the problem solution or information considered important in generating solution elements. Entries may include beliefs, observations, hypotheses, decisions, goals, interpretations or expectations. The system designer may designate a

variety of attributes to entries according to the system's needs. These attributes can include an entry's content, relationship with other entries, its history and other information.

Knowledge sources are independent, event driven processes that produce entries. Each knowledge source is composed of two parts, a condition and an action. The condition describes the circumstances under which a knowledge source can operate. This usually requires the existence of certain previously generated entries. The action of a knowledge source generates new entries or modifies previously generated entries. Knowledge sources operate independently and do not communicate with one another directly. However, they influence each other indirectly whenever the action of one knowledge source generates or changes an entry that satisfies or partially satisfies the condition of another knowledge source.

The blackboard is a global data base containing all entries generated by the knowledge sources during the problem solving process. The blackboard serves two functions. First, it mediates all knowledge source interactions. In this manner, knowledge sources influence one another indirectly by placing and responding to entries on the blackboard. Thus, an entry recorded by the action of one knowledge source may satisfy the condition of another knowledge source. Second, the blackboard organizes all partial and complete solutions generated for the problem under consideration. These solutions comprise configurations of related entries on the blackboard. The blackboard for a particular application will be organized to define important relationships among its entries. Typical arrangements might be specializations for temporal or spatial relationships, domain specific generalizations or other entry classifications. In addition, the blackboard focuses knowledge source activity and thereby is able to improve the system's operating efficiency. Normally, a knowledge source's condition refers to previously generated entries in a particular area of the blackboard, while its action generates or modifies entries in some other area. Knowledge sources need not consider entries in areas of the blackboard not mentioned in their conditions or actions. Consequently, the blackboard structure is a framework for organizing, inspecting and generating entries.

The final component of the blackboard architecture is its control mechanism. During the problem solving process, many different knowledge sources may have their conditions satisfied simultaneously. An intelligent control mechanism determines which of the currently satisfied knowledge sources should execute next. How the execution of knowledge sources is ordered is the job of the scheduler, one of the main components of the control mechanism. The scheduler employs strategies based on the structure of the expert system and the nature of the domain.

B.1.5.3. Truth Maintenance

In a classic logical system, facts that are once true are always true. In the real world, of course, this is not the case---assertions that become true may later become false. The conclusions that one has come to because of a once-true, now-false assertion may themselves no longer be valid. The approach of "truth maintenance" (or "reason maintenance") is intended to provide a bookkeeping method for deciding which conclusions are still valid, and which are not. Essentially, the chain of reasoning that led the system to conclude a fact are stored, so that when belief in an assertion changes, the system can update the conclusions it now believes.

Recently a variety of extensions to truth maintenance systems have been proposed; the most recent series of improvements are detailed in dekleer86a, dekleer86b, dekleer86c, which discuss manipulating assumption sets (rather than single assertions), using truth maintenance for default reasoning, and the role that truth maintenance plays in an overall reasoning system. For earlier discussion of truth maintenance, see doyle79.

Planning systems, and other problem solving processes, often generate a collection of deductions as they proceed. These deductions are the result of computations, previous deductions and new inputs from the environment. These deductions may undergo changes in their validity for a number of reasons, especially because of changes in the environment and its representation. This problem could be particularly acute in the military context because the normal errors associated with sensor readings are compounded by deliberate misrepresentation by the other side. Without some means for maintaining a consistent data base of currently believed deductions, the operation of the system will be seriously degraded. This is the function of truth maintenance.

Truth maintenance systems record and maintain proofs. These proofs are made up of justifications connecting data structures called nodes. The nodes typically represent assertions, rules, or other program beliefs. Nodes may have several justifications, each of which represents a different method of deriving belief in the node. For each node, the truth maintenance system computes whether or not belief in the node is justified by the existence of a non-circular proof from the basic hypotheses and the set of recorded justifications. The set of such non-circular proofs is recorded as the well-founded support of the believed nodes.

When a new justification is recorded by the system, the system checks to see if the new justification can be used to provide well-founded support for some currently unsupported node. If such nodes are discovered, they are marked as believed and the new justification is attached to the node as its well-founded support. The truth maintenance system attempts to propagate this effect among its other nodes and their justifications.

B.1.5.4. Resolution Theorem Provers

The information available for the solution of problems (especially difficult and interesting ones) is usually not complete. This implies that a method of solution must transcend mere search or evaluation and include a capability for deducing new facts from the original set of facts known to the problem solver. A major objective of artificial intelligence is the automatic extraction of these new facts from an existing body of knowledge. Resolution theorem proving, which resulted from Alan Robinson's combination of the resolution principle with automatic symbolic deduction techniques Robinson65 accomplishes this objective, although with definite limitations. The resolution principle operates on data which can be placed in a uniform representation (clausal form) and mechanically deduces new facts, including specified hypotheses, using a single rule of inference (the resolution principle).

Resolution theorem proving met with early success and enthusiasm and continues to be an active area for research today although there are still important constraints on its usefulness. The main difficulty with automatic resolution theorem proving is that the search space generated grows exponentially with the number of facts used to describe the problem and for most problems, this space becomes unpractically large. The use of domain independent meta rules to limit this growth has been largely unsuccessful.

Progress in automatic deduction is currently being made in several areas. One approach is to employ domain knowledge to guide the inferencing. For example, to use domain knowledge to decide when to use forward or backward inferencing (see above). Another approach is to employ higher order or nonclassical logics. Work in these areas is still in progress. Other forms of automatic deduction exist, including nonresolution theorem proving, the Boyer-Moore Theorem Prover, nonmonotonic logics and logic programming (eg Prolog). See nilsson80 for a discussion of various types of resolution.

B.1.5.5. Tableau Method

A variation on the Resolution method, the tableau method, was put forward more recently as a mechanism of automated deduction manna80, manna86 . Instead of representing all facts, goals, and rules in a homogenous data base, there is a partitioned data base that distinguishes between assertions and goals. A variety of rules of deduction are employed to combine facts with facts, convert facts to goals, etc. (in contrast to the single rule used in resolution). While the kinds of transformations that the database undergoes are more complex, the non-homogeneity of the database allows more selective syntactic heuristics to be employed in guiding the search for an answer.

B.1.5.6. State Space Search

Much, if not all, of planning can be considered a search through a state space---if the original condition of the world is presented as a description of an "initial state," and the goal is presented as a "final state," then the planner's job is to search through the

possible states to discover a traversable path from the start to the finish (this depends, of course, on the operators that are available for moving from one state to another). While this outlook provides a high-level framework for evaluating and contrasting planning methods, in does not answer the fundamental question of how the state-space search should be managed. Management of search is of course precisely what classic AI planning systems are attempting to accomplish.

State-space search is a form of problem solving using states and operators. A state is a data structure representing a snapshot of the problem at one stage of the solution. Operators transform one state into another. Planning may be viewed as a search problem. In this formalism a desired state (the goal) is described and the set of possible steps leading from the initial conditions to the goal is designated as the search space. A resulting plan is a sequence of operations leading from the initial state to the goal state. State space searches can be classified as blind searches or heuristically guided searches.

Blind searches are searches in which the potential solution paths are considered in arbitrary order, using no domain specific information to guide them. Several blind search methods are described here, differing mainly in the order in which the states of the search space are investigated. In these cases it is assumed that there is always a method for finding all of the successor nodes of a given node, that the state space graph is a tree and that each node has a link to its parent.

The first blind search method to be considered is breadth-first search. This method expands states in order of their proximity to the initial state. (All states at depth n are processed before going on to depth, n−1 until the first goal state is encountered.) This is an exhaustive search method which guarantees finding the shortest solution path, but not necessarily in the most efficient manner.

Uniform-cost search is a generalization of breadth-first search, designed to find the cheapest path from the start state to a goal state. In uniform-cost search a positive cost is assigned to the paths between each state. The cost to any state then, is the sum of the costs from the initial state to the state under consideration. Solution paths are investigated in order of increasing cost. In the case of identical costs the breadth-first selection mechanism is invoked. When a solution path (plan) to a goal is found it is guaranteed to be a minimum cost plan for the system.

A third blind search method is depth-first search. In depth-first search, the most recently generated node is the next node to be expanded. This expansion continues until either a goal state is reached or no more expansions are possible. In the latter case, the expansion process is resumed at the last unexpanded state junction. This method is not guaranteed to find any goal state, and may continue forever. One safeguard against infinite wrong pathway expansions is to introduce a depth bound, at which depth the system responds as if no more expansions were possible. This eliminates searching infinite dead ends, but also raises the possibility of missing goals

and thus failing to produce any plan at all.

Focus of attention issues---Search Control A key issue in implementation of search-based methods of planning is how to control the "focus of attention" of the system. There are a variety of methods, ranging from the opportunistic blackboard approach of wandering attention, to tightly controlled top-down constructions. If search is considered the major paradigm of planning, search control is really the major concern of planning implementations.

There is a great deal of literature on the subject of search in AI planning. For readable, in-depth discussions of various search techniques, including depth-first, best-first, breadth-first, agenda mechanisms, etc., see [nilsson71, nilsson80].

The blind search methods described above each executed state expansions in an orderly fashion but there was never a sense of actively searching out the goals. If goals were encountered at all it was merely by chance while carrying out the state expansion procedure. In many practical problems, the possible search space is so large and the goal states are so sparsely distributed that goals would not be found in the times available for computation. The purpose of heuristically guided search is to reduce the search effort by using knowledge of the domain to direct the state expansion process. Several examples of heuristically guided search methods are described below. In these examples some combination of the following processes are generally employed: decide which states to expand next, decide which successor states to generate, and identify states to be discarded from the search tree.

The first heuristic search method we will consider is best-first search. In best-first search, the most promising node is always expanded next. Best-first search may act globally on all currently generated, but not yet expanded nodes, or it may be constrained to some subset of these nodes. In any case, there must be a method, called the evaluation function, for determining the suitability of any node as a step in reaching the goal. The magnitude of the evaluation function is inversely proportional to its suitability, hence the node with the lowest value is selected for expansion. (Breadth-first, uniform-cost and depth-first searches are special cases of best-first search, depending on how the evaluation function is defined.) The nature of the problem influences the type of solution and evaluation function. Several are discussed below.

The first problem type is characterized by multiple solution paths with different costs, and it is desired to find the minimum cost solution path. The A* (A-star) algorithm can be used under these circumstances. The evaluation function, $f^*(n)$, is the sum of $g^*(n)$, the estimated cost from the start node to the current node plus $h^*(n)$, the estimated cost from the current node to the goal node. eg

$$f^*(n) = g^*(n) + h^*(n)$$

If $h^*(n)$ is less than or equal to the actual cost to the goal, all arcs have positive costs and are bounded from below, then A^* is guaranteed to find a solution path of minimal cost if one exists.

B.1.5.7. Knowledge representation languages

A key issue in planning, and in fact in all of AI, is the choice of *knowledge representation* that is chosen for the system. There are two key aspects that must be considered. First, the language chosen needs to be **epistemologically adequate**---it must be capable of representing the things that the designer wants to have represented. This involves both the choice of underlying language (frames, first-order logic, etc.), and vocabulary (i.e., what objects will be part of the vocabulary of the system---often called the system's *ontology*). Second, the language needs to be **heuristically adequate**---the system must be capable of deducing the things that the designer wants deduced.

A wide variety of languages have been used for representing knowledge in planning systems. It is not our intention here to provide any kind of comprehensive summary, but rather to briefly mention a few of the competing possibilities. For a collection of important papers in knowledge representation (presenting many of the key approaches), see brachman85 .

Many AI planning systems have opted for an internal knowledge representation language that closely resembles first-order predicate logic [nilsson80]. The advantages of this choice are the clear semantics that logic enjoys, as well as the well-understood (and sound) methods of deduction that can be employed to derive new knowledge from old knowledge. Higher-order logics (meta-logic) and modal logics (dealing with issues such as time) have also been employed, though they tend to amplify one of the most serious problems that the logic approach has---it is usually quite inefficient.

Expert systems have popularized the use of more or less homogenous representation of facts and rules. While these systems can be more efficient than general logic representations, they gain this efficiency by sacrificing expressiveness. In addition, a number of extensions need to be made to straight logic so as to represent issues such as uncertainty, and to control search among the rules hayesroth83 .

Semantic nets, collections of nodes connected by arcs, were an early popular method for representing knowledge in deductive systems maida85 . These relatively unstructured collections of data were later supplanted for the most part by *frames*, structured collections of data that were grouped together to representational advantage minsky85 . Each frame consists of labeled slots that can hold values, or can point to other frames in a hierarchy. Frames allow for natural ways to deal with representation issues such as default values, and allow for natural "inheritance" of attributes from frames higher in the hierarchy (e.g., a dog frame might be an instance of the mammal frame, and we could deduce certain properties of dogs because of things we know about

mammals). As with semantic nets, however, the true semantics of frames are not always clear, and the kinds of deductions that can be made are sometimes similarly imprecise, or implicit in the code that runs the system (difficult to evaluate as to correctness or completeness).

Newer systems have attempted to incorporate hybrids of logic, expert system rules, frames, etc. into a single flexible and powerful system. See, for example, the LOOPS system discussed in stefik83.

References

1 James Allen.
Towards a general theory of action and time.
Artificial Intelligence 23(2):123-154, 1984.

2 Douglas E. Appelt.
Planning Natural Language Utterances to Satisfy Multiple Goals.
SRI International, Menlo Park, California Tech Note:259, 1982.

3 Ronald J. Brachman and Hector J. Levesque.
Readings in Knowledge Representation.
Morgan Kaufman Publlishers, Inc., 1985.

4 J. Carbonell.
Counterplanning: a strategy-based model of adversary planning in real-world situations.
Artificial Intelligence 16:295-329, 1981.

5 Eugene Charniak and Drew McDermott.
Introduction to Artificial Intelligence.
Addison-Wesley Publishing Company, Reading, Massachusetts, 1985.

6 Philip R. Cohen.
On Knowing What to Say: Planning Speech Acts.
PhD thesis, University of Toronto, 1978.

7 P. R. Cohen and C. R. Perrault.
Elements of a Plan-Based Theory of Speech Acts.
Cognitive Science 3(3):177-212, 1979.

8 Paul R. Cohen and Edward A. Feigenbaum (Editors).
The Handbook of Artificial Intelligence.
William Kaufmann, Inc., Los Altos, California 3, 1982.

9 Philip R. Cohen and Hector J. Levesque.
Persistence, intention, and commitment.
Proceedings of the Workshop on Planning and Reasoning About Action The American Association for Artificial Intelligence, Timberline, Oregon, July, 1986.

10 Thomas L. Dean.
Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving.
Technical Report, Yale Univeristy Computer Science Department, 1985.

11 Thomas L. Dean.
Intractability and Time Dependent Planning.
In *Proceedings of the Workshop on Planning and Reaoning About Action*, pages 143-164. The American Association for Artificial Intelligence, Timberline, Oregon, July, 1986.

12 J. De Kleer.
An Assumption-Based Tms.
Artificial Intelligence 28(2):127-162, 1986.

13 J. De Kleer.
Extending the Atms.
Artificial Intelligence 28(2):163-196, 1986.

14 J. De Kleer.
Problem Solving With the Atms.
Artificial Intelligence 28(2):197-224, 1986.

15 J. Doyle.
A truth maintenance system.
Artificial Intelligence 12:231-272, 1979.

16 R. E. Fikes and N. J. Nilsson.
A New Approach to the Application of Theorem Proving to Problem Solving.
Artificial Intelligence 2(3/4):189-208, 1971.

17 J. Jeffrey Finger.
Exploiting Constraints in Deductive Design Synthesis.
Technical Report, PhD thesis, Stanford University, 1986.
To appear.

18 J. S. Rosenschein, M. R. Genesereth, M. L. Ginsberg.
Cooperation Without Communication.
In *Proceedings of the National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, Philadelphia, Aug, 1986.
To appear.

19 Michael Georgeff.
Communication and Interaction in Multi-Agent Planning.
Technical Report, The American Association for Artificial Intelligence, Washington, D.C., August, 1983.
Proceedings of the National Conference on Artificial Intelligence.

20 M. Georgeff, A. Lansky, and P. Bessiere.
Proceedings of the 9th International Joint Conference on Artificial Intelligence.
International Joint Conferences on Artificial Intelligence, Los Angeles :516-523, August, 1985.

[21] M. Georgeff and A. Lanksy.
Procedural Knowledge.
Technical Report, In IEEE Special Proceedings on Knowledge Representation. IEEE, 1986.
To appear.

[22] M. L. Ginsberg.
Possible Worlds Planning.
Technical Report, The American Association for Artificial Intelligence, July, 1986.

[23] Andrew R. Haas.
A Syntactic Theory of Belief and Action.
Artificial Intelligence 28(3):245-292, May, 1986.

[24] Patrick J. Hayes.
The Second Naive Physics Manifesto.
Ablex Publishing Corporation, Norwood, New Jersey, 1985, Chapter 1.
pages 1-36.

[25] Andrew R. Haas.
A Syntactic Theory of Belief and Action.
Artificial Intelligence 28(3):245-292, May, 1986.

[26] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat.
Integrating Text Planning and Production in Generation.
Addison-Wesley Publishing Company, Reading, Massachusetts, Building Expert Systems, 1983.

[27] Edward H. Hovy.
Proceedings of the Ninth International Joing Conference on Artificial Intelligence.
Technical Report, International Joint Conferences on Artificial Intelligence, Los Angeles, California, August, 1985.

[28] Kurt Konolige.
Belief and Incompleteness.
Ablex Publishing Corporation, Norwood, New Jersey, 1985.
pages 359-403.

[29] Kurt Konolige.
Belief and Incompleteness.
Ablex Publishing Corporation, Norwood, New Jersey, 1985, pages 359-403.
Chapter 10.

30 Amy L. Lansky.
A Representation of Parallel Activity Based on Events, Structure, and Causality.
In *Proceedings of the Workshop on Planning and Reasoning About Action*, pages 143-164. American Association for Artificial Intelligence, Timberline, Oregon. Jul. 1986.

31 A. S. Maida and S. C. Shapiro.
Intensional Concepts in Propositional Semantic Networks.
Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985, pages 169-189. Chapter 9.

32 Z. Manna and R. Waldinger.
A Deductive Approach to Program Synthesis.
ACM Transactions on Programming Languages and Systems 2(1)::90-121. January, 1980.

33 Z. Manna and R. Waldinger.
Special Relations in Automated Deduction.
Journal of the ACM 33(1)::27-39. January, 1986.

34 J. McCarthy.
Circumscription-a Form of Non-monotonic Reasoning.
Artificial Intelligence 13(1,2)::27-39. 1980.
reprinted in 'Readings in Artificial Intelligence'.

35 D. McDermott and J. Doyle.
Non-Monotonic Logic.
Artificial Intelligence 13(1,2)::41-72. 1980.

36 Drew mcDermott.
A Temporal Logic for Reasoning About Processes and Plans.
Cognitive Science 6::101-155, 1982.

37 K. R. McKeown.
Generating Natural Language Text in Response to Questions about Database Queries.
Technical Report, University of Pennsylvania, 1982.
PhD thesis.

38 Marvin Minsky.
A Framework for Representing Knowledge.
Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985, pages 245-262. Chapter 12.

39 Robert C. Moore.
A Formal Theory of Knowledge and Action.
Ablex Publishing Corporation, Norwood, New Jersey, 1985, pages 319-358. Chapter 9.

40 Robert C. Moore.
Semantical Considerations on Nonmonotonic Logic.
Artificial Intelligence 25(1)::75-94, January, 1985.

41 Nils J. Nilsson.
Problem-Solving Methods in Artificial Intelligence.
McGraw-Hill Book Company, New York, 1971.

42 Nils J. Nilsson.
Principles of Artificial Intelligence.
Tioga Publishing Company, Palo Alto, California, 1980.

43 Nils J. Nilsson.
Probabilistic Logic.
Artificial Intelligence 28(1)::71-87, February, 1986.

44 R. Reiter.
A Logic for Default Reasoning.
Technical Report 79-8, University of British Columbia Department of Computer Science, Vancouver, B.C., Jul, 1979.

45 Elaine Rich.
Artificial Intelligence.
McGraw-Hill Book Company, New York, 1983.

46 J. A. Robinson.
A Machine-Oriented Logic Based on The Resolution Principle.
J.ACM 12:23-41, 1965.

47 Jeffrey S. Rosenschein.
Synchronization of Multi-Agent Plans.
In *Proceedings of The National Conference on Artificial Intelligence*, pages 115-119. The American Association for Artificial Intelligence, Pittsburgh, Pennsylvania, Aug, 1982.

48 Jeffrey S. Rosenschein and Michael R. Genesereth.
Deals Among Rational Agents.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91-99. The International Joint Conferences on Artificial Intelligence, Los Angeles, California, Aug, 1985.
Also published as STAN-CS-85-1042 March, 1985.

49 Stanley J. Rosenschein.
Formal Theories of Knowledge in AI and Robotics.
New Generation Computing 3:345-357, 1985.

50 Jeffrey S. Rosenschein.
Rational Interaction: Cooperation Among Intelligent Agents.
PhD thesis, Stanford University, Oct, 1985".
Also published as STAN-CS-85-1081 (KSL-85-40), Department of Computer Science, Stanford University.

51 E. D. Sacerdoti.
A Structure for Plans and Behavior.
PhD thesis, Stanford University, 1975.
Also published as Technical Note 109, SRI, Menlo Park, CA.

52 A. L. Samuel.
Some Studies in Machine Learning Using the Game of Checkers.
IBM Journal of Research and Development (3)::211-229, 1959.

53 Yoav Shohan .
Ten Requirements for a Theory of Change.
New Generation Computing 3:467-477, 1985.

54 Yoav Shoham.
Time and Causality From the Standpoint of Artificial Intelligence.
PhD thesis, Yale University, 1986.
To Appear.

55 M. Stefik, D. G. Bobrow, S. Mittal, and L. Conway.
Knowledge Programming in Loops.
AI Magazine 4(3)::3-13, 1983.

56 G. J. Sussman.
A Computer Model of Skill Acquisition.
American Elsevier, New York , 1975.

57 R. Waldinger.
Achieving Several Goals Simultaneously.
Tioga Publishing Company, Palo Alto, California, 1977, pages 94-136, Chapter 8.
Reprinted in Readings in Artificial Intelligence;editor Bonnie Lynn Webber and Nils J. Nilsson, editors;

58 Bonnie Lynn Webber and Nils J. Nilsson (Editors).
Readings in Artificial Intelligence.
Tioga Publishing Company, Palo Alto, California, 1981.

59 Robert Wilensky.
Planning and Understanding.
Addison-Wesley Publishing Company Reading, Massachusetts, 1983.