

THE THEORY AND USE OF SCENARIOS

RAJENDRA SOOKDEO WALL

University of Massachusetts
Amherst, Ma.

COINS Technical Report 82-31

THE THEORY AND USE OF SCENARIOS

A Dissertation Presented

By

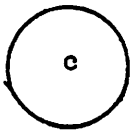
RAJENDRA SOOKDEO WALL

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1983

Computer and Information Science



Rajendra Sookdeo Wall

All Rights Reserved

This material is based upon work supported by
The National Science Foundation
under grant #IST-8017343

Any opinions, findings and conclusions or recommendations
expressed in this publication are those of the author and do not
necessarily reflect the views of the National Science Foundation.

THE THEORY AND USE OF SCENARIOS

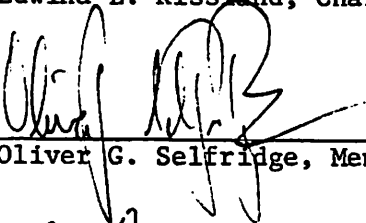
A Dissertation Presented

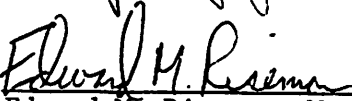
By

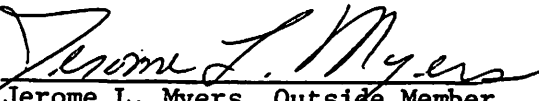
RAJENDRA SOOKDEO WALL

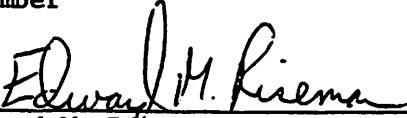
Approved as to style and content by:


Edwina L. Rissland, Chairman


Oliver G. Selfridge, Member


Edward M. Riseman, Member


Jerome L. Myers, Outside Member


Edward M. Riseman, Department Head
Computer and Information Science

DEDICATION

To the Memory of my Mother and Father

ACKNOWLEDGEMENT

I would first like to thank my committee: Professor Edwina L. Rissland, for her help in sorting out the wheat of my ideas from the chaff; Professor Oliver G. Selfridge, for keeping me honest; and Professors Edward M. Riseman and Jerome L. Myers for their help and advice. I would also like to thank Professor Caxton C. Foster for getting me through my Master's degree and started towards this degree.

I would then like to thank my family: my wife Jilynn for keeping me going, and my sons Viran and Davin for putting up with me.

I would also like to thank all the rest of the COINS community for providing a rewarding research atmosphere, and in particular, Daniel D. Corkill for his help and advice on my work, giving me access to his reference library, and developing CLISP, the language that the system is written in.

Finally I would like to thank Dennis Wang, President of the University of Massachusetts Strategic Games Club, for his help as Tactical Expert.

ABSTRACT

The Theory and Use of Scenarios

February 1983

Rajendra Sookdeo Wall,

B. S., University of Nebraska/Lincoln

M. S., University of Massachusetts/Amherst

Ph. D., University of Massachusetts/Amherst

Directed by: Professor Edwina L. Rissland

This thesis examines the problem of the generation and evaluation of future world states in complex planning domains. A tool to aid plan formation in such domains, especially those containing uncontrolled agents, is presented. The tool is called a scenario: a collection of projections of future events, each of which is based on a set of assumptions about the behaviors, intentions and effects of the various processes involved. By examining the scenario generated for a planning problem the planner will be able to judge the potential futures and choose a course of action.

The scenario is generated by performing categorical analysis on the planning problem under consideration: The total set of possible future states is in effect divided up into a set of classes or categories. This division is on the basis of the goals of the

planner, the problem situation, the courses of action available to the planner to achieve the goals, and assumptions about the activities of the uncontrolled processes. For each category one or more examples are chosen to illustrate the kinds of events that could occur in that class of future world.

Tree search of future states is avoided completely by generating the examples by retrieval and modification of experience. A knowledge base of previous experience in this domain is provided by an expert. When an example of a future world class is needed a memory of an experience with a similar problem is recovered from the knowledge base. The experience is modified to fit the current situation exactly and then used to illustrate the potentials of that future world class.

This thesis then applies these ideas to two domains: errand running and conflict simulation games. A domain independent Tactical Assistant was built to perform categorical analysis and retrieval and modification of experience. Experiential knowledge bases were built for the two application domains to give advice on small tactical problems.

TABLE OF CONTENTS

ACKNOWLEDGEMENT v

Chapter

I. INTRODUCTION 1

 A. Purpose of Scenarios. 2

 B. Review of the Issues. 3

 C. What Was Proposed 4

 D. What Was Done 5

 E. What Was Discovered 6

 F. Outline of the Rest of the Document 7

II. DISCUSSION OF SELECTED RELATED WORK 11

 A. Planning Systems in Conventional Domains 11

 B. The Problems of Uncertainty 17

 C. Foreign Processes 23

 D. Cognitive Science 30

 E. Summary 34

III. THEORY OF SCENARIOS 36

 A. What Is a Scenario? 36

 B. Creating a Scenario 50

 C. A Computational Solution - the Tactical Assistant 54

 D. Glossary of Scenario Theory 57

IV. SCENARIOS IN PRACTICE - A SIMPLE DOMAIN 60

 A. The Errand Running Domain 60

 B. Use of Scenarios to Solve the Problem 63

 C. Planning Errand Running by Simulation 63

 D. By Retrieval and Modification of Experience 85

V. THE USE OF SCENARIOS IN A COMPLEX DOMAIN. 93

 A. Introduction to Conflict Simulation Gaming 93

B. Why Examine the Conflict Simulation Game Domain?	96
C. Using Scenarios to Solve the Problem	102
D. Overview of the Constrained Scenario Generation System	107
E. Human Study	113
F. Examples of the System	123
G. The Knowledge Base	139
H. System Details	147
I. Support Software	151
VI. RESULTS, CONCLUSIONS AND IMPLICATIONS	152
A. Results from the Errand Running Domain	152
B. Results from the Conflict Simulation Game Domain	159
C. Conclusions and Implications	163
D. A Potential Application	173
E. Future Work	175
REFERENCES	178
ANNOTATED BIBLIOGRAPHY	187
APPENDIX	191

LIST OF ILLUSTRATIONS

1. The Method of Categorical Analysis	42
2. The Structure of a Scenario	49
3. The Local Community	62
4. Output from the Simulation Based Tactical Assistant	81
5. Output from the Retrieval and Modification of Experience Based Tactical Assistant	86
6. An Example Model Experience	91
7. A Portion of the Mapsheet from <u>Chickamauga</u> . .	101
8. Problem 1	115
9. Problem 2	116
10. Problem 3	118
11. Problem 4	119
12. A Relevant Experience for Problem 1	124
13. The Modified Experience	125
14. A Modified Experience from Problem 2 (turn 1) .	126
15. A Modified Experience from Problem 2 (turn 3) .	127
16. An Alternative Possible World Category from Problem 2 (turn 1)	128
17. An Alternative Possible World Category from Problem 2 (turn 2)	129
18. A Helpful Possible World Category from Problem 3	131
19. An Alternative Possible World Category from Problem 3 (turn 2)	132
20. An Alternative Possible World Category from Problem 3 (turn 7)	133
21. An Alternative Course of Action from Problem 3	134
22. An Example Course of Events from Problem 4 . .	135
23. An Alternative Course of Action from Problem 4	136
24. Another Alternative Course of Action from Problem 4 (turn 1)	137
25. Another Alternative Course of Action from Problem 4 (turn 2)	138
26. Detailed Structure of the Knowledge Base . . .	140
27. The Scenario Generation Mechanism	148
28. The Call History from Generating a Scenario . .	192
29. The Call History from Displaying a Scenario . .	195

CHAPTER I

INTRODUCTION

Most domains can be represented by a collection of facts called a world state. Problem solving can often be viewed as a transformation of some current world state into a desired world state known as the goal state. The problem is the difference between the current and the goal state. The solution is a sequence of operations or actions that will accomplish the transformation.

Research in Artificial Intelligence and other fields has developed methods for representing domains, describing problems and finding solutions. To this body of knowledge I suggest adding the use of scenarios to outline the possible futures dependent on what means the planner uses to try and solve the problem and how the other uncontrolled agents behave, before an attempt is actually made to solve the problem.

A. Purpose of Scenarios

In many complex domains full scale detailed solutions to problems are not possible or practical. This could be due to the inherent complexity of the domain as in oil well analysis or medical diagnosis; elements of uncertainty and chance as in errand running or weather prediction; actions of agents uncontrolled by or unfriendly to the protagonist process as in chess or GO; or a combination of all of these factors as in conflict simulation games or raising children.

The desired solution is a plan and a way to implement the plan: a detailed series of actions to deal with or solve the problem, and the order in which to apply the actions. In less complex domains, like the AI "blocks world", it is easier to produce a detailed plan of action. Every single action, its consequences and side effects, can be plotted out and taken care of; a list of actions can be produced that some effector process can then carry out.

Sacerdoti has suggested that in complex domains complete plans are not feasible [SACE79]. Dockery has noted that the task of providing a complete military analysis is impossible for all but the simplest problems; for command and control problems, the combinatorics of situation projection nearly defeat any solution at the outset [DOCK82b]. Rather than attempt to produce such detailed solutions, I propose an alternative of creating a scenario: an

abstract plan outlining the choices of courses of action the planner can undertake, the behaviors and actions of uncontrolled processes, and their consequences. This scenario can then be used to decide courses of action, outline contingency plans or force reexamination of goals set in the problem statement.

Scenarios can be produced by an expert system called a Tactical Assistant that aids planning in domains where complete solutions are unobtainable. This thesis will present the theory of scenarios, their methods of generation, and examples of their use.

B. Review of the Issues

The first issue that this work deals with is planning and problem solving in complex domains with uncontrolled foreign processes. Previous work in this area is detailed in Chapter Two, and the solution by the use of scenarios in Chapter Three. Trying to make plans in these domains involves complex knowledge and reasoning, and thus complex knowledge structures and reasoning abilities. Most previous work in this area has concentrated on trying to examine the possible futures of a planning problem at the same time as choosing the way to act to produce the desired future. The approach of this work is to separate these two processes; to generate the possible futures for examination and present them as a scenario to the planner who decides the best way to act.

A second issue is the choice of knowledge representation used to encode the various types of knowledge needed to create scenarios. This includes comparison of the merits of retrieval and modification of experience versus rule-based paradigms.

A third issue is the investigation of the different methods of generating scenarios and the knowledge required for each method.

C. What Was Proposed

This work set out to propose, explore and defend an alternative to the planning regimens of the past. The goals were to develop a theoretical background for scenarios and to test the theory first on the simple domain of errand running and then on the more complex domain of conflict simulation games. This was to be done by building Tactical Assistants that used scenarios to aid the formation of plans.

A simultaneous goal was to examine the retrieval and modification of experience paradigm of knowledge representation and use, and its relation to the more familiar rule-based paradigm.

The research work evolved the methodology of Categorical Analysis, which is the foundation of scenarios. This idea was to be tested by building a Tactical Assistant that used a form of rule-based simulation to create outlines of hypothetical plan executions of errand running.

A domain independent system to generate scenarios by retrieval and modification of experience was to be built. It would first be used in the conflict simulation game domain as a tactical assistant to solve small tactical problems, and then in the errand running domain to examine the relationships between the two knowledge representation paradigms.

D. What Was Done

The theoretical basis of scenarios as an aid to planning and problem solving is explained in Chapter Three.

A Tactical Assistant for the errand running domain that used simulation to generate scenarios was built. A discussion of its embodiment of the theoretical ideas is in Chapter Four.

A Tactical Assistant for errand running using retrieval and modification of experience was built and is also discussed in Chapter Four.

A Tactical Assistant for a Conflict Simulation Game using retrieval and modification of experience was built and is discussed in Chapter Five.

E. What Was Discovered

Some of the highlights of this work include: Scenarios were found to be a useful contribution to planning and problem solving, and Categorical Analysis is a useful tool for reducing the complexity of domains containing foreign processes.

The use of retrieval and modification of experience as a form of expert knowledge representation was found to be useful. It allows statements about the future to be made without performing tree search. The search is instead through the knowledge base of experiences, and thus is bound by the size of the knowledge base, rather than a potentially unbound operator space.

The continuation of this work is to apply these ideas to other complex domains (e. g., command and control, emergency room medicine, or geological analysis), as well as to perhaps reexamine those domains for which the current approach is the tree search planning methodology. These results and implications are discussed in more detail in Chapter Six.

Future directions include The Commander, an expert that would act as a complement to the Tactical Assistant: a strategic assistant that would understand issues such as "surprise" and "indirection". Further research is also discussed in more detail in Chapter Six.

F. Outline of the Rest of the Document

Chapter Two reviews some of the related previous work. First, planning systems in non-complex, non-uncertain domains, for example, GPS, STRIPS, ABSTRIPS, HACKER, INTERPLAN, and NOAH, as well as work done in the Errand Running domain are reviewed. Second, the problems of uncertainty, and work done as solutions are discussed: The Executing Robot, Dynamic Worlds, Non-planning Planning, Simulation, and Utility Theory. Third, work in domains with foreign processes is presented: Game Playing, Counterplanning, Statistical Modelling. Finally, relevant work done in Cognitive Science is reviewed.

Chapter Three discusses the theory of scenarios and presents an application, the Tactical Assistant. It first explains what a scenario is, introduces some terminology and presents the methodology of Categorical Analysis. It shows how this procedure can be used to handle foreign processes. The structure of a scenario is then presented. The methods of creating scenarios are discussed: Static Analysis, Simulation, and Retrieval and Modification of Experience. Finally, the concept of a Tactical Assistant is introduced and discussed; what it is supposed to do and how it does it.

Chapter Four discusses some particular implementations of these ideas in a simple domain, errand running. First the errand running domain is presented and the problems involved in it are discussed.

Then the use of scenarios to solve the problem is introduced. Two systems were written to examine errand running: one used simulation to generate scenarios, the other retrieval and modification of experience. First the simulation system is discussed - what it did, how it was implemented and an example of its work. Then the system that performed retrieval and modification of experience is discussed. An example is shown, but the discussion of the implementation is deferred to Chapter Five, since that is where the application for which the system was really intended is discussed.

Chapter Five presents the domain of Conflict Simulation Games and the system to generate scenarios from retrieval and modification of experience. First is an introduction to Conflict Simulation Gaming - what is a conflict simulation game and why it is a difficult domain in which to solve problems; then there is an introduction to the game used in this work, Chickamauga. The features of scenario theory that allow a potential solution to the problem of planning in this kind of complex gaming domain are then discussed: categorical analysis, retrieval and modification of experience, and the Tactical Assistant. The Constrained Scenario Generation System is then introduced, and its workings and abilities discussed.

The human study that was performed to judge the worth of the system is then discussed: the problems, the types of subjects, the suggested solutions, and the comments and reactions of the subjects. Examples of the system are shown. The internal details are then discussed: the knowledge base, its organization, and a simple functional overview. The implementation details are left to the Appendix.

Chapter Six presents the results and conclusions of this work. First the results from the errand running domain are discussed: the first test of the practicality of scenarios, their use in a tactical assistant, the use of categorical analysis, and the two methods of scenario creation implemented. The results from the conflict simulation game domain are then discussed: the conflict simulation game Tactical Assistant, the need for retrieval and modification of experience, the use of categorical analysis, the knowledge base, and the human study.

The conclusions and implications of this work are then discussed: the use of scenarios, a comparison of the two methods of scenario generation, knowledge representation, the constrained scenario generation system and the retrieval and modification of experience paradigm that is its basis, the handling of foreign processes, and the use of categorical analysis to analyze the future.

Finally, possible future work in the area is presented. First, a possible application of this work to Emergency Room medicine is discussed. Then some of the work that could be done in another year of work or by a Master's candidate extending this work is listed. Finally, future directions in research to pursue the implications of this work and extend it to other planning domains are listed.

C H A P T E R I I
DISCUSSION OF SELECTED RELATED WORK

This chapter discusses some of the work that has been done in Artificial Intelligence and other fields that relates to this thesis, and shows why some method of aiding planning in uncertain domains with foreign processes is needed.

First, previous work in conventional domains is examined. Second, systems that attempted to deal with environmental uncertainty. Third, systems that deal with foreign processes. Fourth some of the relevant work that helped aid the formulation and justification of the ideas in this thesis is discussed. Finally, a summary is made of the works that most influenced this thesis.

A. Planning Systems in Conventional Domains

Traditionally, in past works the problems were usually straightforward; for example, to build a tower of blocks with the blocks in a specific order. The world could be specified as a simple state description consisting of wff's - well formed formulas of world fact.

There were specific operators that would perform perfect actions and produce exact results. Sequences of operators could be strung together to accomplish goals. A goal was expressed as a world state description. The main problem for these systems was how to best organize the sequences of operators into a plan to transform the world from the initial state to the goal state.

A.1. GPS

GPS [ERNS69], [NEWE72] used an operator/difference table to select which operator to use. This table was to be the only item of domain specific knowledge in the system. The system would notice a difference between the initial state and the goal state; this difference could be alleviated by the application of the operator indicated by the table.

This operator in turn could require a certain world state before it could operate; a difference between the initial state and this state would imply more operators and so on. This process was called backchaining. The process of noticing differences between the initial and current goal states and selecting operators to resolve those differences was called means-ends analysis.

One problem with this method was when there were more than one relevant operator per difference. Then the selection of the correct sequence of operators became a blind search over the space of relevant operators. Solutions to this search problem occupied researchers for a long time [NILS71].

A.2. STRIPS

The STRIPS system [FIKE71] was different in operation and composition. The world state was again a set of wffs. The operators were described by three lists: a precondition list, an add list and a delete list. The precondition list specified what items had to be true in the world before the operator could act. The add list was the list of items that became true after the operator completed. The delete list was the list of items that were removed from the world state after the operator was finished.

Instead of backchaining, STRIPS used forward-chaining, which meant the search through the operator space was a tree search starting with the initial state and working towards the goal state. The selection of operators during the search was not bound to knowledge about what the operator could accomplish in the same restrictive tabular sense as GPS, but the system could peruse the precondition, add and delete lists of the operators to give it a handle on what to try next.

The system was also capable of storing partial solutions in abstract form for use in later problems. For example, if the system discovered that to put block A on B it needed to perform the sequence [(CLEARTOP B), (PUT A B)] it could save the sequence as the solution to putting block 'x' on block 'y' as [(CLEARTOP y), (PUT x y)], and then reuse the solution with other blocks inserted for 'x' and 'y'.

A.3. ABSTRIPS

Such methods were found to be inadequate for more complex tasks. Sacerdoti developed the ideas of abstract plans in the ABSTRIPS (Abstraction Based STRIPS) system [SACE74]. This system was similar to the STRIPS system except that criticality values were placed on the conjuncts of the operator preconditions. The higher the value the more important the condition. For example, the "go-thru-door" operator might give a high value to the precondition "there must be a door in the room", slightly lower to "the robot must be next to the door" and lower to "the door must be open". ABSTRIPS could set a threshold at a high level and consider only preconditions of that high level as plan constraints, and construct a skeleton plan. The threshold could then be lowered slightly and more constraints considered. Finally a detailed, low level plan would be produced.

A major philosophical change in this work from previous work was the recognition that general problem solvers such as GPS or STRIPS would never be able to solve really difficult problems, and that a structure with domain intensive knowledge was necessary. In fact, ABSTRIPS was dependent on the domain knowledge of the order and assignation of criticality values to prevent it from making incorrect operator choices and having to backup to a higher level in order to fix them.

A.4. HACKER

Sussman developed the idea of "almost correct" plans that would then be debugged [SUSS75]. The domain was again that of the blocks world. The system would first look for a solution in its data base of old solutions, and if there was no complete solution, it would try to write a new one. The new solution would then be examined for known types of bugs by critics. The solution was then tried. If there were any bugs they would be patched and the system would try to generalize the bug so it could be recognized in the future. The solution would then be saved for later use.

A.5. NOAH

In 1977 Sacerdoti presented a revised version of the ABSTRIPS system called NOAH (Nets of Abstraction Hierarchies) [SACE77]. This system also used plan hierarchies and a partial ordered operator sequence to make only necessary commitments. One difference was that the developing plan was kept in a procedural network that made problem detection and correction easier. It used critics, similar to those of Sussman, to provide a global view of the plan and notice problems.

A.6. Errand Running

A multi-level knowledge-source/blackboard model was proposed by Hayes-Roth [HAYE79] to plan the running of errands. The model has as its base a cognitive study showing that when humans plan errand running they rarely do so in a methodical straightforward manner. Rather they seem to seize key events and decisions and fix them in the plan, building the rest of the intermediate plan around them. Eventually a complete plan is formed. The basis for this work is an examination of the HEARSAY processing scheme [LESS77].

Errand running was also examined by Wilensky in the context of using meta-planning to aid goal conflict resolution and conservation of system resources [WILE80]. The system uses projection to debug plans, a process similar to search. The proposed plans may come from

stereotyped solutions, editing of previous plans or the creation of novel solutions. The planner has three parts: a Proposer that suggests plausible plans, a Projector that builds hypothetical worlds from the execution of the plan using simulation, and a Revisor that edits plans when problems are discovered during the simulated execution. The "meta" goals lead to tactics such as to perform all tasks in a given area before going on to the next area.

B. The Problems of Uncertainty

A very troublesome feature of many problem domains is that of uncertainty. This uncertainty can be due to processes with random natures and effects, operators with multiple outcomes, or incomplete information about the true state of the world or the intentions of uncontrolled processes. In this work I treat these problems as differing facets of the same problem, but in most previous work they were considered separate.

B.1. The Executing Robot

The issue of how to plan and act in uncertain environments was investigated by Siklossy & Dreussi [SIKL74]. They propose an executing robot to operate in a domain with inconsistencies. There are two worlds in the domain: the robot's internal idea of how the world looks and the "real world" which is how the world actually is.

When presented with a task the robot makes a tentative plan based on its internal world (not a complete solution to the problem). It then begins to try to execute the plan. It is able to detect inconsistencies between what it expected and the way the world actually is. It then either revises the plan or replaces it with another plan. It is able to search the environment for missing items (e. g., wander about from room to room). There is only a single process operating in the environment, but a malevolent programmer can move things about before the system is started.

B.2. Dynamic Worlds

Chien and Weissman examined a dynamic world in which random processes operated and could change features of the world independent of the planning process [CHIE75]. Their approach was a system similar to that of ABSTRIPS: criticality values were used to generate a plan outline. The lower detail levels were assumed to be solvable one way or another once the system got to that point in its execution. If not the system did not have a readily accessible alternative plan.

McCalla and Reid looked at a simulated robot taxi driver named ELMER that faced dynamic road hazards such as other cars, red lights, and pedestrians [McCA82]. The robot was given the task of trying to get from one point in a simulated city to another. The knowledge it had about the city was in the form of a map as well as records of all

the previous trips it had taken. To plan a new trip it tried to splice together various pieces of knowledge about previous trips it had taken. This knowledge is collected after each trip and saved as a generalized and abstracted record of the trip.

The main idea of this work was not the handling of dynamic processes (which was done by demon-like procedures) but to implement a complete model of planning: creation of plans, execution of plans, and "learning" from the results of execution.

B.3. Non-planning Planning

The idea of non-detail-intensive planning is taken to the extreme by McDermott who proposes effectively doing no planning at all, rather just doing what you can now and trying to reschedule disrupted tasks at some later time [McDE78]. The Scheduler must have some domain knowledge in order to make reasonable choices of when to reschedule what task. What has happened is that if the system can get anything done at all it is due to some planning ability of the Scheduler, otherwise such a system could get into trouble if faced with future dire consequences that must be avoided now if at all.

B.4. Simulation

Wesson in his Air Traffic Control system handles uncertainty by running a detailed discrete simulation to see what the future holds [WESS77]. The domain is that of a system trying to direct the flight paths of airplanes through a fixed airspace so that collisions are avoided. The uncertainty in this domain is the exact paths of the airplanes being directed. The simulation is a discrete instant by instant projection of where the planes would go.

If at any time the system notices that two or more planes have been projected to intersect the same place at the same time it resets the simulation time to an earlier time, tells one or more of the intersecting planes to change course, and then runs the simulation forward again to see if that change solved the problem.

Since decisions and their consequences are included in the simulation the system is able to backtrack and correct potential errors "before they occur". In one sense this approach is similar to a tree search in which the goal state is a future with no collisions. Since the situation is always changing with the introduction of new planes the system is only as good as the speed of the search.

An example of an event driven simulation can be seen in the work by Lowrance and Freidman [LOWR77]. This work built a system that was able to model: 1) simultaneous processes; 2) processes characterized by a continuum of gradual change; 3) involuntarily activated processes; 4) time as a continuous phenomenon.

The system uses what the authors call scenarios, which are each a description of one process containing three types of information: 1) the conditions for initiation, 2) the effects this process will have on the state of the world, and 3) the conditions under which the process continues. In effect, the scenario in this work is an abstract description of the way this process will behave in the future.

B.5. Utility Theory

Sproul proposes using utility theory [LUCE58], [LUCE64] to create a probabilistic model that can calculate the payoff of a plan [SPRO77]. The domain is defined with each operator having a probability of success and a value of payoff if successful (and perhaps a negative payoff if failure). The final goal is also given a value. The search of the operator space is similar to that of STRIPS. In addition the model can be used to judge the worth of trying to resolve uncertainty in the environment.

One first generates utility and probability functions for all strategies. Once this is done then apparently "incomparable" strategies can be compared. The use of utility functions automatically handles uncertainty.

At each "instant" of plan generation a three element choice is available: to plan, to look (examine the environment to eliminate uncertainty), or to act. Each has cost and utility functions. The decision of which to do is made by a Scheduler aided by upper and lower bounds for the cost and utility functions. This allows it to decide numerically which choice to make.

A discussion of this work leads to minmax theory [vonN53, LEIN68]. In both cases excellent theoretical results can be obtained, but whether those results can be maintained in practice is open to question. One reason is that in the real world the numbers chosen for values and probabilities will always be approximations to the "real" values for those numbers, if such values can be said to exist at all. These numbers could be based on statistical surveys, and using statistical methods their possible error could be reduced. Other implementation and combinatorial problems arise when there are multiple complex choices that can be made on either side. The main objections arise when dealing with behaviors and tactics of foreign processes, which will be discussed below.

Minmax theory is also the basis of the Alpha-Beta search technique used in searching the game trees of conventional games [NILS71].

C. Foreign Processes

A foreign process is any activity that takes place in the environment that is uncontrolled by the planning process. In the Terminology section of chapter three I discuss the definition and classification of foreign processes in more detail. In general, foreign processes cause changes to the world state that may postpone or prevent the achievement of the goals of the planner.

Foreign processes are difficult for the systems of previous work: first, because of the problem that actions are not equivalent to permanent world state changes, especially as the system plans further and further into the future; second, because of the problems of guessing the intentions or behaviors of foreign processes; and third, because of the inherent complexity of the domains in which foreign processes operate.

Most solutions have involved simultaneous completion of two tasks: the generation of possible future world states for evaluation and choosing a course of action that the system should pursue to achieve the desired future state. In most cases this solution has

involved some sort of search of the future state space.

This search consisted of a projection of the effects of processes's actions on the world state. From the initial state an action by one process would produce a new state. Then the action of another process would be considered, producing a third state and so on. Each time the planning process had a "turn" at making a change to the world state the action would be to try and guide the world state to the desired goal state.

If there were more than one choice of action in a given state then the method of deciding which action to take would be based on the evaluation of the possible worlds produced. The more numerous and complex the actions, the more difficult the choice. In addition, if the domain involved unfriendly or competitive processes, the choice of action by those processes would be based on an unknown, their evaluations which would depend on their own goals.

Various methods were proposed for the combinatorial problems associated with search [NILS80]. Some of these dealt with various means of removing some of the branches from the search tree, others with differing methods of traversing the tree, still others with new ways to evaluate the possible worlds generated.

C.1. Game Playing

Work has been done in domains that must deal with potentially disruptive foreign processes. Most of this work has been in game playing. One of the first game playing programs was Samuel's checker player [SAMU63]. It used a polynomial to evaluate the game board position and suggest moves. It is an extraordinarily powerful player, but the best humans can beat it consistently.

Work on Chess has taken two paths. The first was an attempt to play the game heuristically [HARR74, BERL72, BERL73] which did not meet with much success. The second was to use the speed of the computer to simply perform a brute force search of the game tree. This approach is the one used by the highly ranked programs of today.

Slate & Atkin developed Chess 4.5 which uses high-speed computing power to examine the game tree in detail [SLAT77]. They use a "plausible move generator" to do some pruning, but once that is done a standard tree search is performed.

Wilkins developed a system that was sort of a combination of the two approaches. Although his system eventually searches the game tree, it first uses knowledge about chess to limit the breadth of the search by creating a plan to refer to during the search [WILK79]. Thus the search is much deeper and fewer branches are explored than with a blind or purely minmax (alpha-beta) Chess game tree search.

Despite its apparent complexity. Chess is actually quite straightforward in its play. Only one piece can be moved by a side at a time, any piece can destroy any enemy piece, there are no terrain effects, normally only two pieces can engage in combat at a time, there is no combat at a distance, and both players have complete information. This appears to be the major reason why brute force search is able to accomplish anything. In a more complicated game such as a conflict simulation game, a tree search would be overwhelmed after only one or two ply.

So what should be done in a more complicated domain in which tree search is combinatorily prohibited? As will be shown below, when using the Retrieval and Modification of Experience method of scenario generation examined in this work, knowledge about the domain is used to avoid all game tree type search. Thus all the computational difficulties of tree search are avoided.

The game of GO has been worked on by numerous researchers, including Rietman & Wilcox. Their INTERIM.2 system uses multi-level pattern analysis and recognition to identify the board situation, weigh the various moves available and choose a sequence of moves to make. This system tries to avoid examining the game tree in detail due to the overwhelming cost [REIT79]. GO is a domain rich in image processing potential. It may be shown in the future that with enough knowledge about which patterns imply which correct lines of play,

competent GO playing programs will be developed that will not need to perform search.

The game of Backgammon was investigated by Berliner. He proposes doing away with symbolic heuristics altogether and instead using a polynomial hill-climbing algorithm to guide play [BERL80]. The polynomial can be tuned to improve its play, and expressions can be added to factor in special situational values when certain features are detected. The main importance of this work to the thesis is that it examines a domain with both random processes (the dice) and unfriendly processes (the opponent).

One of the problems with this approach is that it does away with, or ignores, all the semantic detail available in a domain. Berliner's position is that semantic knowledge is too discrete to properly capture all the nuances of goal pursuit. This may be true when performing tree search in machine play, but it is clear that when humans play more complicated games they use domain intensive semantic knowledge. In addition, hill-climbing does not lend itself easily to long range planning.

Mostow examined playing the card game Hearts. Hearts is a multi-player game and although each player tries to win individually, at times certain players must cooperate [MOST79]. The uncertainty in this domain is the exact position of the unseen cards, and there are

foreign processes present (the other players). The approach taken is to set up bodies of tactical rules in the form of a production system and then to use that production system to suggest plays. The main impetus for this work is to examine the boundaries of the production rule/knowledge source paradigm. Planning as such is handled by rules that incorporate knowledge about possible long term goals and make local suggestions accordingly.

C.2. Counterplanning

The POLITICS system of Carbonell also deals with foreign processes in the form of adversaries. He has developed a series of domain independent heuristics for performing counterplanning - planning to disrupt or impede an opponent's goals [CARB79a, CARB79b]. These heuristics are such things as "to protect your goals, threaten the enemy's high valued goals". Although many of these seem to be merely common sense, it is useful to see them written down.

The main purpose of this work is to examine human beliefs and conflict resolution when humans have conflicting goals such as in political decision making. For example, what to do if the Soviets build more submarines (build more US subs? Call for arms controls?).

C.3. Statistical Modelling

The Quantified Judgement Model of Dupuy is applied to statistics of military battles to predict the outcomes of those battles. The set of statistics is very large and detailed, and the model usually gives good results when used on historical data. When the prediction is incorrect, the problem is usually due to incorrect assignment of intangible variables such as morale, initiative or surprise, rather than to the equations of the model [DUPU79]. The model is an example of what we will later refer to as static analysis. A difficulty of this approach is the acquisition of correct values for the numbers to be fed into the model. This is the same kind of objection raised to minmax above, which will now be pursued in more detail.

In the case of determining an enemy's behavior and tactics, one would have to base the numbers on past performance; e. g., in ten similar situations in the past this commander did tactic-w four times, tactic-x three times, tactic-y twice, and tactic-z once. If probabilities are based on these numbers (and tactical decisions projected from these results), what happens if the enemy commander knows you are basing your actions on a probabilistic analysis of his past behavior? Any change could render the old statistics meaningless.

Dockery proposes using fuzzy set theory to aid military decision making [DOCK82a]. Fuzzy set theory is used to apply probabilities to data acquisition and planning, including potential analysis of commander behavior.

The problems for military information systems include:

1. They are employed and are expected to perform their most demanding support tasks in the presence of fatigue and extreme user stress.
2. They contain data which become worthless after a very short period of time, are often imprecise and sometimes simply wrong.
3. The amount of raw data available is overwhelming
4. They are susceptible to the input of deliberately wrong and misleading information into the data base.
5. They are targets of destruction and deliberate disruption.

D. Cognitive Science

Experiential knowledge has been represented by scripts [SCHA77]. Scripts were devised to aid in understanding natural language descriptions of situational action. The description of action may not be complete, and some points must be inferred. For example, in a restaurant situation, the description might be "Katrina went to the

restaurant. She ordered beef chow mein. She left the tip under the teapot and went home". The restaurant script would contain a description of the events that normally take place in a restaurant. From this the understanding system could infer such things as: Katrina sat at a table, she ate her meal and she paid for her meal.

Scripts were later incorporated (in a completely different form) in the memory structure called MOP's [SCHA80], which forms relationships among similar experiences so that they can be used by a process that may encounter situations similar to those in the MOP. MOP's involved a complete revision of the theory developed for scripts. It was decided that scripts were not flexible enough to describe all the things involved in real world experiences. The main function of MOP's is to describe activities rather than plan activities.

Work investigating the epistemological issues involved in representing knowledge by examples was done by Rissland. In addition, work has been done using examples as a data base in a system called a "Constrained Example Generator" [RISS78, RISS80]. The system would accept a query as a list of constraints. An attempt would be made to retrieve an example that met the constraints exactly. If none could be found, a "close" example would be chosen and modification attempted, the hope being that an intelligent modification could produce the desired item. This procedure is referred to later as the

retrieval and modification of examples paradigm. The relationships among the database examples, routines to judge closeness and perform modification are a matter of ongoing research.

The other domain of interest is that of conflict simulation games. Planning in this domain is somewhat similar to that done by actual military planners. In a series of articles, Pratush has examined the relationship between game playing and reality [PRAT81]. Formal military planning is outlined in field manuals (e. g., FM-101-5 [USAR69]).

Analysis has been done of actual military planning in a controlled situation by Hayes-Roth. This study categorized the cognitive strategies used by planners and the effectiveness of those strategies in terms of the correctness of of the plans produced [HAYE80]. The three strategies discovered turned out to be very similar to those used by people who play conflict simulation games [WANG80]. The strategies are: analysis of problem specific information, mental simulation and retrieval of past experience. The categories are similar to those used in the generation of scenarios [WALL81].

Historical analysis of plans and outcomes of military conflicts are discussed in the two volume work by Esposito [ESPO59]. In general this work approaches each conflict in a manner similar to that of

planning analysis. First, a top level view of the situation is presented (the campaign level); this shows the goals of the forces involved on all sides and the general plan of each force to achieve their goals. Second, each engagement in the campaign is examined; this shows the goals and plans in more detail. Finally each battle in the engagement is detailed, showing the plan and the results: what happened to the goals of each side.

The military often run highly complex war game simulations as training (e. g., FM-105 [USAR67]). These simulations are run with humans playing the parts of the high level commanders and the computer handling low level details such as logistics. From these exercises it is hoped the trainees will learn the manipulation of the strategic elements involved.

Part of the conceptual basis for scenarios comes from the rationale behind conflict simulation games [SIMO80]. One of the reasons people play games involving historical simulation (aside from the enjoyment of the mental challenge) is they allow investigation of the question "what if?" Players can examine the effects of different strategic goals, command decisions, tactical choices, troop placement, force strength, etc, on the outcome of the conflict. Most published games codify these variations to the historical situation in a section of the rules called "scenarios".

E. Summary

The following references most directly influenced the work in this thesis. First, the ABSTRIPS system of Sacerdoti [SACE74] was important. Besides changing to a domain specific problem solving approach the idea of a hierarchical plan was intriguing. At what level do people normally plan? The issue of a detailed plan versus a simple plan is raised. Should a system always make a detailed plan? In what sorts of domains would detailed plans be impractical or pointless? In looking for the answers to these questions an investigation of foreign processes was begun, eventually leading to the current work.

Second, the Executing Robot of Siklossy and Dreussi [SIKL74]. This work began investigation of a number of complex real world problems, such as planning in an environment in which the world model is out of date, reconciling the internal world with the real world, and how to cope with an external agent (the programmer) whose actions are uncontrolled. What happens if uncontrolled changes can be made while the plan is being executed? Subverting goals that have already been finished? What if the uncontrolled agent can be directly affected by actions of the planner? These questions lead to the investigation of gaming domains where they can be examined.

Third, pure numerical game theory, for example the text by Leinfellner [LEIN68]. Minmax theory is very elegant and has impressive proofs, but what are its limits? In what sort of domain would it have problems? Is there a possibility for a more symbolic, non-numerical approach to game-playing? What knowledge, other than probabilities, etc, would be needed to play games? How would that knowledge be represented for use by a machine? These questions led to investigation of the various Artificial Intelligence paradigms for planning and knowledge representation.

Finally, the work on representing the knowledge about a domain by having a set of examples of that domain, done by Rissland [RISS78]. The continuation of this work, the Constrained Example Generation paradigm [RISS80] appeared to be the kind of model needed for a planner trying to handle very complex domains. It fit well with intuition about the way people think and plan when playing conflict simulation games. What was needed to extend the paradigm to planning problems? Could it be used to represent and manipulate the knowledge needed to play conflict simulation games? Could such an extension be used for any given complex planning domain? These questions led directly to this thesis.

C H A P T E R I I I
T H E O R Y O F S C E N A R I O S

This chapter introduces the ideas on which the thesis is based. First, explanation of what a scenario is, and why it is needed. Second, some terminology is introduced that will be used in the succeeding sections. Third, the conceptual foundations of scenario theory are discussed: categorical analysis and how it is used to handle complex phenomena such as foreign processes, course of action selection, and possible world examples. Fourth is a discussion of the methods by which scenarios are created. Finally, the Tactical Assistant, an expert system that uses scenarios to aid user planning is introduced.

A. What is a Scenario?

A scenario is collection of projections of future events, each of which is based on a set of assumptions about the behaviors, intentions and effects of the various processes involved. Each projection depicts the course of events in this interpretation of the future - the important events or actions taken by the various processes involved.

Stated another way, the total set of possible future world states is in effect divided up into classes or categories on the basis of assumptions about the processes involved. From each class one or more examples are chosen to illustrate the kinds of events that could occur in that class of worlds.

As has been shown above in chapter two, previous attempts to solve the problem of planning in domains with foreign processes have considered the tasks of examining hypothetical future worlds indistinguishable and simultaneous from the task of choosing a course of action to pursue that would produce the desired future world. The approach taken by scenario theory is that domains with foreign processes are so complex that the two tasks must be separated; that only after the higher level planning process has gotten an idea of the range of future possibilities can complex strategic issues such as "surprise", "initiative", or "indirection" be considered.

The goal of this approach then is for the scenario to provide for the user or the higher level planning process enough information about the future so that a reasonable decision about the correct course of action can be made.

A.1. Terminology

In each domain where scenarios are used there is some goal or set of goals (the goal-set) to be secured. These goals consist of statements about the desired state of the world under consideration. They can be items to be achieved, maintained, or avoided. A set of goals can contain interrelated or even contradictory statements.

Each goal has a value assigned by the process or user generating the goal; that value reflects the worth of the goal to the assigning process. For example, in the errand running domain, the value of the goal of buying a newspaper might be low while that of meeting a dentist appointment when you have a toothache might be very high.

The world state at the time the goals are chosen is called the situation. The situation can include not only the current status of the processes involved in the world, their positions and activities, but also any past activity that has led to the current state.

The task of taking a given situation as transforming it into the goal is called the problem. The combination of the goal-set statements and the situation description is also referred to as the problem.

In any domain such as errand running or conflict simulation games, there are numerous tactics available with which to secure goals. Each tactic describes an action or set of actions designed to meet a particular goal or subgoal. Any goal may have numerous applicable tactics. The set of applicable tactics may change depending on the situation. Any particular tactic may affect more than the designated goal favorably or adversely. Additionally, in complex domains the tactics are not guaranteed to produce a solution (e. g., due to actions by foreign processes).

Thus for each goal in a given situation a tactic or set of tactics can be chosen. A set of tactics to achieve all of the goals in the goal-set and the order to apply them in is called a course of action. For any given set of goals there may be different courses of action available, each with advantages and disadvantages.

A listing of the way a possible future is envisioned, with all actions by all interested processes included, is called a course of events. This listing may be thought of as a "series of snapshots" showing the state of the world as it undergoes changes. The activity resulting in the creation of a hypothetical course of events is called a projection.

A foreign process is any activity or action that can cause changes in the world state that are uncontrolled by the planning process. These may include potentially disruptive processes whose actions may prevent accomplishing friendly goals. In this work, foreign processes are grouped into three categories according to their intentions:

- o A random process is one that occurs without any discernable reason. Usually it is an environmental process unrelated to any goal or purpose.
- o An unfriendly or inimical process is a process whose goals are always in conflict with friendly goals and whose actions can always be interpreted as hostile. Often unfriendly processes have goals that are in direct conflict with friendly goals, i. e., the goal of the unfriendly process is to prevent the accomplishment of friendly goals. This is the case in most gaming domains.
- o A competitive process is one that may sometimes have goals that conflict with the friendly processes' goals. Usually conflicts arise from common desires for domain resources. However, it is possible for competing processes to cooperate to accomplish common goals; additionally, such a process could operate for a period of time without revealing how its actions should be interpreted.

A.2. The Method of Categorical Analysis

Using scenarios to create projections of the future is done differently from the search methods normally used to examine the future. Instead of generating actions, responses to actions, responses to responses and so on, I skip all world state search completely by a five-step process (illustrated by figure 1):

1. The parameters concerning uncontrolled processes under which future worlds can be distinguished are selected by an expert familiar with the domain. These scenario parameters are facets of the uncontrolled processes that can impact the achievement of the planner's goals. For example, in the errand running domain such a parameter would be the effects of random processes; in the conflict simulation domain, the behavior of enemy processes.
2. Possible values for the scenario parameters are selected by the domain expert that will break the range of a parameter into a discrete number of classes or categories; e. g., in errand running the values for the random effects would be "non-interfering" (everything goes right), "neutral" (things likely to go wrong do so) and "detrimental" (everything goes wrong); in conflict simulation the behavior of enemy processes might be "aggressive" or "passive".

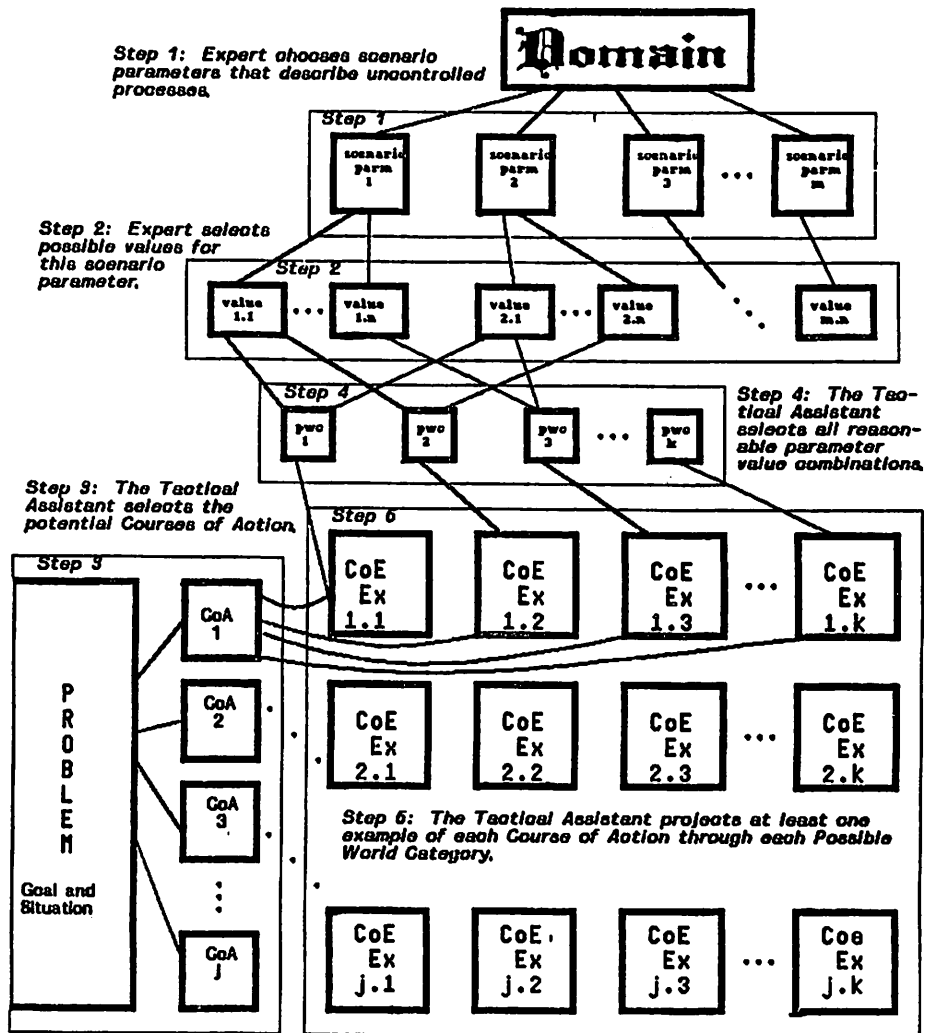


Figure 1. The Method of Categorical Analysis.

3. Given the current goals of the process, the current situation in which to achieve those goals, and knowledge of the tactics applicable in the domain, the Tactical Assistant program (see section C, below) can generate a set of the possible courses of action.
4. The Tactical Assistant chooses each plausible combination of scenario parameter values. Each combination is called a possible world category and all action or operator sequences that would result in such a world are considered identical (in other words, how you get there doesn't matter as much as where you are); for example, if there are two parameters, random-effects and enemy-behavior, one possible world category could be (neutral aggressive), and another might be (detrimental passive).
5. For each possible choice of course of action, for each possible world category, the Tactical Assistant generates at least one example to demonstrate what could be the potential course of events. This generation of a course of events example for a given course of action for a given possible world category is the process referred to as projection. The set of examples for a course of action given a possible world category is called a segment, and the collection of all the segments for all the courses of action is called the scenario.

Step one, the selection of parameters, is made by the domain expert on the basis of the goals of the process. The chosen parameters are facets of the world whose differing values would make a difference to the achievement of goals. For example, a facet of an errand running domain might be the color of the car used to run the errands; however, it would not be chosen as a scenario parameter since it has no bearing on the accomplishment of errand running tasks. On the other hand, the amount of gas in the car (perhaps an unknown at the time of planning) could be.

Step two, the selection of values for the parameters, is also done by the domain expert. The values are selected so that the set of them "spans" the range of the parameter. They represent assumptions about that facet of the possible future world the parameter is concerned with. The use of assumptions allows the consolidation of a variety of future worlds into a single class by considering all worlds in which that assumption is true to be the same.

The third step of the segmentation, the course of action selection, is done by the Tactical Assitant, a program with domain specific knowledge of the tactics that can be pursued to achieve the stated goals in the current situation. The suggestions of tactics are based on abstract situational descriptors rather than exact knowledge of what course of action is really good or worthwhile. At this point the courses of action suggested are only that - suggestions - and it

will be seen in the examples produced for each world class which ones would have what result.

By "plausible choice of parameter category combination" in step four, I mean those choices that are reasonable in combination as well as reasonable given the problem goal and situation. When parameters are interrelated, a choice of value for one might limit the reasonable choices for another; for example, choosing the behavior of an inimical process to be "aggressive" might preclude the choice of more passive defensive tactics for its course of action. This selection of possible world categories is done by the Tactical Assistant using its knowledge of the domain.

For step five, the methods of generating the examples used to demonstrate the possibilities of each course of action for each possible world category, are detailed below in section B, Creating a Scenario. They are by static analysis, simulation, and retrieval and modification of experience. Domain specific knowledge in various forms is needed for each method. Deciding how many and which examples of each category are needed to fully explain that category is problem specific.

The complete scenario includes at least one example of the future world each course of action would produce given each reasonable assumption. By presenting only examples of each assumption class, the

important features of the various future worlds are highlighted while avoiding the details of innumerable tree search branches.

The selection of possible world parameters and their values must be made by someone able to decide what are the relevant features and the reasonable assumptions about those features for the domain in question. There are few guidelines for making this selection as unfortunately this portion of knowledge engineering is still an art. Some of the guidelines that have been helpful so far have included:

- o Whenever deemed important to the possible completion of goals the effects of foreign processes are noted. These effects must be organized in some way such that a parameter can be established for them.
- o Actions by foreign processes that impede achievement of high value goals are considered more important than those for low value goals.
- o Actions by random processes with high probabilities of occurrence are more important than those of low probabilities.
- o Harmful actions of competitive processes are more important than neutral actions.

- o Actions of unfriendly processes to fulfill their high valued goals are more important than those to fulfill their low valued goals.

A.3. Handling Foreign Processes with Categorical Analysis

One of the main advantages of the use of categorical analysis is its ability to encompass the effects of uncontrolled processes. In order to make a projection of the future in domains with foreign processes, assumptions must be made about their behaviors and desires. These assumptions are done categorically. For example, in one possible world, the behavior of an unfriendly process might be assumed to be aggressive and neutral in another. Or, the effects of a random process might be assumed to be helpful in one possible world and detrimental in another.

Once given a set of assumptions, foreign process actions can then be postulated. These actions are then interpreted in light of their effects on the proposed course of action under consideration. A complete scenario consists of an analysis of each course of action under all reasonable assumptions about foreign processes. The "sequence of snapshots" that make up a course of events example is the result of this course of action analysis.

Random processes are assumed to be non-intelligent and undirected. No motives are ascribed to them and thus they are not truly considered to have a "behavior". Instead the activities of these processes are considered in light of what their actions will do to the accomplishment of goals. This is done by a possible world classification category called "random effects". The values this parameter takes may vary from domain to domain, but in general are things such as "helpful" - denoting worlds where random events are beneficial, "inconsequential" - the effects can be ignored, or "detrimental" - where everything goes wrong. This can be thought of as a threshold on which probabilities of random events are measured; those events with a higher probability than the threshold are considered to occur.

Categorizing competitive and inimical process behavior is a method of abstracting one of the great unknowns in these domains, the exact goals of uncontrolled processes. The assumption is that the behavior of a competitive or inimical process will change depending on whether or not its own goals are at stake in the conflict area. Thus without actually determining what the competitive or inimical process's goals are, with this approach one can make statements about the kinds of things the process might do.

A.4. Summary of Scenario Theory

This, then, is the basis of scenarios: In response to a set of goals, a set of courses of action to achieve the goals is proposed. Consideration is then given to the effects of foreign processes. This consideration is along parameters whose values will represent assumptions about the effects, behaviors and intentions of the foreign processes involved. At least one projection of the future course of events that could be expected given these assumptions is the generated as an example of that possible world category.

A complete scenario consists of an example that is an analysis of each course of action under a representative sample of reasonable assumptions about foreign processes. The course of events example consists of a "sequence of snapshots" that make up a projection of future activity. A scenario is thus a set of analyses of courses of action within assumptions about foreign processes.

A.5. The Structure of a Scenario

The basic outline of the structure of a scenario is as follows:

- I. Course of Action 1. Set of tactics chosen to accomplish the goals within the limitations of the given situation.
 - A. Segment 1.1. Possible World Classification Parameters and their values:
 - Classification of effects of, or thresholds for, random processes.

Behaviors and intentions of competitive or unfriendly processes.

- i. Projection 1.1.1. Example of Course of Events given these assumptions.
 - a. Tactics and their expected consequences.
 - b. Foreign process effects.
 - ii. Projection 1.1.2. Example of Alternative Course of Events given these assumptions.

. . .
 - iii. Projection 1.1.3. . . .
- B. Segment 1.2. Analysis of first course of action based on the next set of reasonable assumptions.
- i. Projection 1.2.1. Example of Course of Events given these assumptions.

. . .
 - ii. Projection 1.2.2. Example of Alternative Course of Events given these assumptions.

. . .
- C. Segment 1.3. . . .
- II. Course of Action 2. A different way of achieving the goals. Segments and Projections analogous to I.
- III. Course of Action 3

Figure 2. The Structure of a Scenario

B. Creating a Scenario

The fundamental part of producing a scenario is the generation of the projections used to exemplify the consequences of each of the courses of action within each of the possible world categories. There are three methods of creating these examples:

1. static analysis
2. simulation
3. retrieval and modification of experience.

B.1. Static Analysis

Static analysis means looking at the current situation and having enough knowledge about the domain to immediately draw conclusions about future events. Static analysis can also be used to a limited extent in more complex domains, such as predicting outcomes of battles [DUPU79] given a large amount of data.

Although static analysis has been identified as one of the methods used by humans when planning, in a sense it is a form of simulation to return immediate results. It can be thought of as taking the values of the various situational parameters, plugging them into a large equation, and calculating the result. This result is some number representing the kind of outcome to be expected.

Knowledge about the domain is expressed in the assignments of the various factors and weights in the analysis equation. A different assumption about a foreign process would mean plugging in a different value for one of the weights in the equation. There is thus no search when using static analysis.

B.2. Simulation

Simulation involves examining the current situation and using knowledge about the ways in which processes function in the domain to project the results of the operation of those processes on the current situation. This requires a model of the world: what things can change and how, as well as a model of each active process - what it can change, when, how, and in the case of foreign processes, why. Most game playing systems use a form of search of the simulation space to decide which move to make. A good example of simulation is Wesson's Air Traffic Control system [WESS77].

The selection of a possible world category means, for random processes, the setting of a threshold against which the probabilities of the corresponding random events are tested. Whenever such a probability exceeds the threshold it is assumed to have occurred. For inimical or competitive processes it means changing the weighting factors used to select their actions.

In either case, the simulation of the hypothetical course of events is focused on the goals of the planning process - the actions taken to achieve those goals and the possible interference caused by the foreign processes.

Knowledge about foreign processes and their effects would probably be best kept in the form of rules. Alternatively, some sort of schematized approach like that of Lowrance and Friedman [LOWR77] could be used.

B.3. Retrieval and Modification of Experience

The last method, and perhaps the most interesting, is retrieval and modification of experience. This involves having a large body of experience in the domain: When a new situation is encountered the data base of experience is examined and a situation similar to the current one is recalled. Along with the remembered situation is knowledge about what happened when certain things were tried - what worked, what didn't, etc. The experience is then an example used to illustrate the current situation and possible futures. This is similar to the Constrained Example Generation method of Rissland on which it is based [RISS80].

Differences between the remembered situation and the exact current situation indicate modifications or interpretations that must be made in order to comprehend the full implications of the experience. These modifications produce the projections that make up the scenario.

The selection of values for the various possible world category parameters means the formation of constraints. These constraints are then used during the examination of the knowledge base to select the relevant experiences.

The knowledge of the experience base would be represented by frames or schemas. All the features and facets of the memory would be in a single structure.

C. A Computational Solution - The Tactical Assistant

This section introduces a means of using the theory of scenarios in a problem solving aid, called a Tactical Assistant.

C.1. Use of Scenario Generation by a Tactical Assistant

A Tactical Assistant is a form of expert system designed for users who have high level strategic goals, but who are not immediately concerned with the low level details of a domain; it will help them choose a specific course of action.

The user proposes a set of goals to be achieved within the current situation. The Tactical Assistant takes this set and generates a scenario analysis of the courses of action that may achieve the goals, and presents it. In response, the user may change the goal set or redefine the situation (e. g., allocate different

resources to different goals). The Tactical Assistant then generates a revised scenario. This iteration continues until the user is satisfied with the outline of the future presented (or gives up).

The Tactical Assistant is given the parameters with which to distinguish future worlds and their possible values. It contains domain specific situation analysis and tactical expertise routines that it uses to select courses of action. It then uses knowledge about the domain to choose the reasonable possible world categories for this problem. Finally it must generate a projection of each course of action through each possible world category - the result being a course of events example to be presented to the user.

In the errand running domain, this process can be thought of as the user first making a list: "These are the things I would like to do today". The Tactical Assistant, with its knowledge of the town, traffic patterns, etc, generates a scenario laying out the ways to accomplish the goals given different tactical choices, noting any conflicts or problems that may arise. This scenario is reviewed by the user, who then may reschedule, postpone or cancel proposed actions, in turn causing the Tactical Assistant to revise the scenario. The system through its analysis can alert the user to possible problems. This analysis is the equivalent of asking a series of "What if?" questions, for example, "What if I run out of money?" or "What if I get stuck in traffic?"

In the conflict simulation gaming domain, the user posts a series of objectives for the friendly forces to achieve. In addition, suggestions or specifications about force disposition and dispersion, times of goal completion, and tactics can be made. The Tactical Assistant examines the knowledge base of experience for appropriate past experiences. These are modified to fit the current situation and are then presented to the user for review. Again, this review may cause modification of objectives, suggestions or specifications, in turn generating new scenarios, etc.

The Tactical Assistant could also aid the user during execution of the plan. If something went wrong it would still be able to suggest one of the alternative courses of action to recover and continue towards the goal.

C.2. How Does the Tactical Assistant Work?

The Tactical Assistant as an expert system has the detailed knowledge about the domain that must be used in making predictions. This knowledge includes the tactics available in the domain, a detailed model of the domain, and detailed models of foreign processes operating in the domain. The knowledge may be in the form of rules for simulation, frames of experience, or both.

The method of generating projections of hypothetical courses of events is dependent on the complexity of the domain, the kind and quality of expert knowledge available, and the amount and kinds of new knowledge to be added as the system is used.

From the goal set and the current situation the Tactical Assistant develops the plausible courses of action. Since any particular tactic may affect more than its designated goal, the Tactical Assistant must understand and be able to handle such side effects. The Tactical Assistant must examine the available courses of action, weigh the costs and benefits, note any possible problems with foreign processes, and summarize the projected courses of events as a scenario for the user. Figure 27 in Chapter Five discusses the implemented Tactical Assistant.

D. Glossary of Scenario Theory

The following is a list of the terms used in scenario theory and a brief description of each.

course of action - a tactic or set of tactics designed to accomplish a given set of goals in a given situation.

course of events - a sketch of the way things could turn out if the course of action is followed and the assumptions about the actions of foreign processes are correct.

foreign process - any domain element that can make changes to the world state outside of the direction of the planner.

foreign process behavior - intelligent or directed foreign processes are thought to have reasons for doing what they do. Although inferring exactly why a foreign process does something is an extremely difficult problem, its attitudes can be postulated.

goal - the objective of the planner. It expressed as a statement or series of statements about the world state. This collection of statements is also called the set of goals or goal-set.

possible world category - a set of assumptions about the future effects of the uncontrolled elements of a domain.

problem - the task of getting from the current world state described in the situation to the goal state desired by the planner.

process parameter - a feature of the domain that may affect solving the problem.

process parameter value - a label for a specific mode or behavior for the process parameter.

projection - the process of examining a course of action within the assumptions of a possible world category and generating a course of events example.

scenario - a collection of sketches of the possible futures of a domain.

segment - a collection of sketches of the possible futures of a domain given a choice of course of action and a set of assumptions about the future.

situation - a description of the current world state.

tactic - a list of the actions to take to accomplish something in the domain.

C H A P T E R I V

SCENARIOS IN PRACTICE - A SIMPLE DOMAIN

This chapter covers the use of scenarios in a simple domain - errand running. Scenarios were generated to outline outcomes of errand running tasks by two methods. One was a conventional rule based simulation. The second was Retrieval and Modification of Experience using the Constrained Scenario Generation System. Thus not only was the use of scenarios tested, but also the methods of generation could be compared.

A. The Errand Running Domain

When working in the errand running domain one is given a set of goals, a physical space to perform in and a set of events that can occur randomly to disrupt the execution of actions. The task is to lay out a course of action that will accomplish the goals with a minimum of effort.

The goals in the errand running domain consist of a specific task description, such as "get groceries" or "cash paycheck", some idea of when the task should be accomplished, and how valuable this goal is. The task description implies other information such as where to go to accomplish the goal (a grocery store, a bank) and the object of the

goal (food, money).

The time specification of the goal can be vague (e. g., "some time this morning") or exact ("appointment at 2:30"). The value of the goal gives an idea of how much consideration should go into scheduling actions to meet the goal and into worrying about events that could disrupt such actions.

The actions used to accomplish goals are for the most extent directly related to those goals. An action is scheduled to move to a location where the goal could be accomplished (e. g., a shopping center with a supermarket) followed by the action to perform the task (e. g., buy the groceries).

The physical space the actions are performed in is the local community: Where the shopping centers, etc, are in relation to "home" and each other, what shops does each contain, traffic conditions for various area, individual store peculiarities, etc. The random events or elements are those unsuspected things that can occur when trying to accomplish the goals. They are such things as the store being sold out of the item you want, the item costing more than you expected, or your running out of money on a long shopping trip, or being delayed by heavy traffic on the way to an important appointment. These events may cause rescheduling of tasks, extra trips or unwanted delays. Figure 3 shows a map of the community.

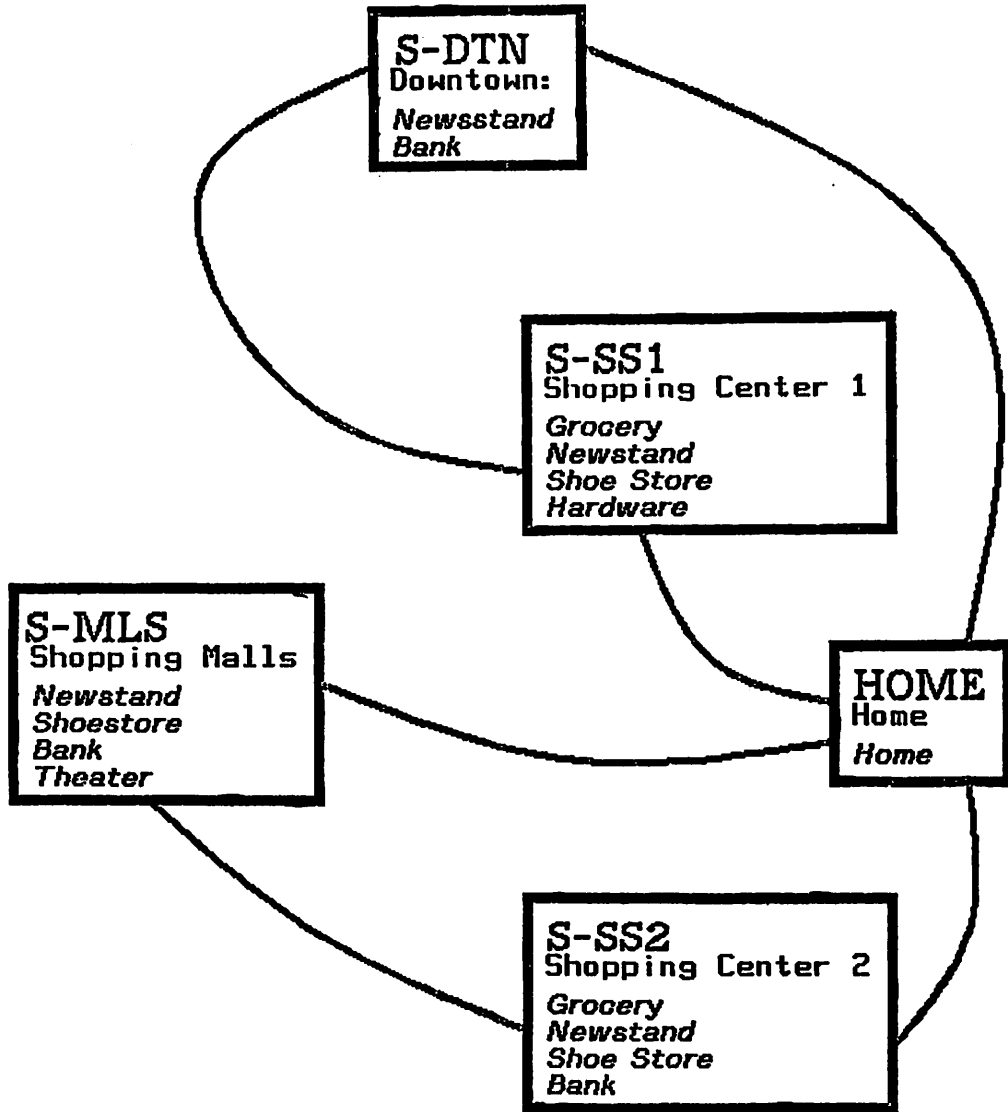


Figure 3. The Local Community.

B. Use of Scenarios to Solve the Problem

Although the Errand Running domain is a simple one, it shows the use of scenarios. It also demonstrates with a single simple parameter the working of Categorical Analysis: the choosing of values for a parameter based on uncontrolled processes, those of random events encountered during shopping.

The choice of a single parameter to reflect the assumptions about all random processes is done to minimize the cost of system effort as well as to model the normal feeling of most people when they are frustrated by the various subproblems, that is, most have a single feeling that "things are going wrong" rather than carefully monitoring each process, yet at the same time each process is distinct in its effects.

In both versions of scenario generation, no situational parameters were included. Situational parameters could have included weather, special events (e. g., holiday weekend) or something similar.

C. Planning Errand Running by Simulation

A Tactical Assistant was implemented to aid in planning errand running trips in the local community by generating possible world examples by simulation. In particular, the implemented system was able to do the following:

- o Take an arbitrary list of goals consisting of things to be accomplished.
- o Accept values rating the importance of each goal.
- o Accept specific times of when the tasks are to be performed.
- o Model three different random processes: heavy traffic, desired item sold out and insufficient funds for purchase.
- o Include conditional probabilities in random process models as functions of space and time.
- o Allow activation of concern about random processes to be determined by threshold.
- o Generate courses of action to accomplish the goals.
- o Implement the errand running tactics: meet-appointments, highest-value and minimize-travel.
- o Generate courses of action by using the tactics in a hierarchy: meet-appointments, highest-value, and minimize-travel.
- o Summarize courses of events as possible world projections.

- o Use scenarios to highlight possible goal conflicts or random effects.
- o Show most convenient course of action first.
- o Track usage of pertinent resources (e. g., money).
- o Track space and time relationships among goals to allow easier comparison of courses of action.
- o Save the alternative courses of action for use as needed during execution of the suggested course of action.

The tactics used in the simulation method of generating projections of the future were represented as rules in the form of LISP COND expressions. The tactics that could be exemplified using the graph structure generated by the goal analysis were minimize-travel, fulfill highest-value, and meet-appointments. The rules were structured to allow a hierarchy of tactical choice, i. e., order the actions to meet the requirements of tactic B given that all demands of tactic A were met. The implemented system was ordered: (1) meet-appointments, (2) highest-value, (3) minimize-travel.

The tactics represented planning rules such as "leave home an hour before the specified completion time of the goal", "try to perform as many high valued goals as possible", "given a choice

complete the goal at the location closest to home", "try to accomplish as much as possible in each trip", and "if possible, avoid heavy traffic areas".

C.1. Random Elements

There were three random elements implemented:

1. Heavy traffic. Heavy traffic could occur while the errand runner was travelling from one place to another. If it occurred then the traveller would be delayed. Included in the heavy traffic process model was knowledge about parking difficulties in the various areas of town.
2. Item sold out. Item sold out was a random event that could occur at any store other than the bank. It meant that the store no longer had in stock the item the user was looking for.
3. Insufficient funds. Insufficient funds could be thought of in two ways: one, that the price was more than the user was willing to pay. The other, that the price was more than the user was able to pay.

Each random element model was represented as a frame or association list. The slots in each random element frame were:

- * The name of the element.
- * The action of interest.
- * The action estimation function.
- * The initial estimate pairs.
- * The conditional factor list.
- * The function expressing the effects of the element.

The value of the name slot was simply the name of the element. The action of interest referred to the type of action that this element could affect, e. g., heavy traffic affects movement actions.

The action estimation function was the name of a function that when applied to the action would return a value to be used to determine the initial estimate and the conditional factors.

The initial estimate pairs slot was filled with a list of pairs. Each pair consisted of a name and a percentage value. The name was matched against the value produced by the action estimation function to determine the initial probability estimate. For example, for the random element item-sold-out the action estimation function was the

location of the goal - what kind of a store it was. This value was then used to find a match in the initial estimate pairs to give a first rough guess of how often that store would be sold out of that product. It was at this point that it was specified that a bank would not run out of money.

The conditional factors slot was also a list of pairs. The first element of the pair was a conditional expression concerning the world state, the time of day, the errands performed so far, etc. The second element was a list of conditional change pairs similar to that of the initial estimates in that the first item was a name to match the action estimation function but the second item was a function to be evaluated. The result of this function was the conditional estimate.

The random element effects slot was filled by a function that would be evaluated to determine the results of the action of this random element on the course of action.

Use of the random element models consisted of taking each proposed action in the course of action and "applying" each random process to it. This "application" meant first that the random element's action of interest was checked against the type of the action under consideration. If there was a match it meant the element could affect this action and further concern was necessary. Thus the random elements were similar to Sussman's critics in HACKER [SUSS75].

Next the initial estimate was made by first applying the action estimation function to the action and then using the result to find a match in the list of initial estimate pairs.

The third step was to check for any changes in the probability estimate due to conditional factors. This was done by evaluating the first element, the conditional expression part of each conditional factor pair. If it evaluated as True then the list of conditional change pairs was checked.

The list of conditional change pairs was checked for a match of the first element of the pair to the result of the action estimation function. When a match was found the second item of the pair was evaluated. The result was the conditional factor estimate and was added to the initial estimate to give a total percentage value of the probability of the random event taking place.

This percentage probability was then checked against the given threshold of concern. If it was greater then the event was considered to have occurred and its effects must be reckoned with. This was done by evaluating the random effects function of the random element. All results were applied immediately.

C.2. Implementation of the Simulation Method

The implemented system is a four phase process. First is goal definition and tactic initialization. Second is action conflict recognition and resolution. Third is the actual scenario organization. And fourth is the presentation to the user.

C.2.1. Phase 1 - Goal Definition and Tactic Initialization

During the first phase of processing the user is asked to define the goals of this run.

- The object of the goal must be specified e. g., "food" for the goal "get groceries".
- The value - a worth rating from one to ten.
- The desired time of completion of the goal.

The time of completion can be specified as either a suggested time or a mandatory time. Specification of a mandatory completion time forces the system to accomodate that goal at exactly that time, while a goal with a suggested time will be fit in wherever convenient as close as possible to the suggested time. Mandatory completion times are thus appointments to be met.

The system then examines the tactics available to accomplish each goal. These are then instantiated in a graph structure using the graph processing language called Grasper [LOWR78]. Linkages are then made among the actions of the tactics noting the following relationships: Spatial, temporal, value and conceptual.

Spatial links are made between any actions that could take place in the same area, for example, all the actions that could be done in one of the shopping centers. This will aid later phases that use errand running convenience heuristics in clustering actions. Links are also made between actions that could take place in adjacent sectors. Such links are used to order actions to minimize travel time.

Temporal links are made to express six different relationships between two actions on the basis of their desired times of completion. The first three relationships consider time specifications that are exactly the same: the first is between two actions that have the same mandatory times of completion. This is an irresolvable conflict that must be handled by special means during phase 2. The second is between a mandatory time and a suggested time. This link alerts later processes that the suggested time action will have to be done earlier or later. The third type of temporal link is between two suggested time actions. One or the other (or perhaps both, depending on other links) will have to be performed at another time.

The last three temporal links are similar, but are between actions that have time specifications "near" each other (within an hour as presently implemented). So again we have mandatory-mandatory, mandatory-suggested and suggested-suggested links. These links allow the system to check for indirect conflicts that may be caused by travel time from one sector to another.

Value links are made to allow the system to try to accomplish as many high value actions as it can once all the appointments are met. The links represent a partial ordering among the actions based on value, in effect sorting the actions.

Conceptual links are made between nodes that are related by some implementation designated concept. The implemented concepts that were linked were such things as use of money, type of action (movement, purchase) and object of goal. For example, an action of buying a newspaper would be linked to an action of going to the bank, since each involved a monetary transaction, even though one used up money and the other replaced it.

This complete graph structure is then saved for use by the second phase.

C.2.2. Phase 2 - Conflict Recognition and Resolution

During the second phase of processing the system uses the linkage network set up in the first phase to sort out incompatible actions: those that required being at two places at the same time. Such a requirement was noted by the class of temporal link between actions with mandatory accomplishment times that were the same or too near each other.

Processing began by examining all the actions to meet appointments in order of decreasing value. A subspace was created and the first such action placed in it, along with all other potentially compatible actions. Only actions with direct conflicts with this action were left behind. Note that the potentially compatible actions were compatible with the initial high valued action; and that they were not necessarily internally compatible with each other.

If there were any actions left, a second subspace was created and the next highest value mandatory accomplishment time action was placed in it, along with all of its potentially compatible actions. This process of creating subspaces and filling them with actions continued until the original set of action had been completely examined. If there had been no appointments in the beginning then all the actions would have been placed in the first subspace.

Now each subspace was examined for internal conflicts. If any appointment time conflicts existed, the subspace was split into two new subspaces each with a copy of all the non-conflicting actions and one of the actions in conflict. These new subspaces were then placed on the list of subspaces to be checked for internal conflicts.

Eventually a set of subspaces was produced that had no internal conflicts. Each subspace contained a set of actions that should be compatible with each other. The next step was to combine them into a course of action.

C.2.3. Phase 3 - Course of Action Organization

Tactics have been outlined to accomplish each goal. Incompatible actions from those tactics have been identified and the conflicts resolved. At this point the actions have not been finalized. This means, for example, that an action to go buy groceries has been selected but which grocery store to patronize has not.

This phase of processing examines each subspace produced by phase 2 and constructs an explicit course of action from it by instantiating course-of-action links among the actions. Due to memory limitations only the course of action representing the use of the chosen tactic hierarchy was actually linked; the other alternative courses were only outlined. Recovering explicit detail of alternative courses

requires running a set of routines to reorder the tactic hierarchy and re-running this phase.

The examination of each subspace begins by looking for actions to meet appointments. The highest valued such action is chosen to be one of the first anchor points. The other initial anchor point is the action "first" which is merely to be at home to begin a day of running errands.

The high valued action is checked to see which of its choice of location instantiations has the highest valued action cluster. An action cluster is made up of all the actions of non-mandatory completion time that could be accomplished in that sector. The value of an action cluster is the sum of the values of the actions in that cluster. Thus each location in which the high valued action could be performed is checked to see what other actions could be performed in that same sector "at the same time". Those other actions are clustered and their values added up. The location with the highest valued cluster is the one chosen for performing the high valued action. If two or more clusters have the same value then the location that minimizes travel time is chosen.

This ordering: choice of location by minimization of travel to highest value cluster, given the meeting of appointments; is an example of the tactic hierarchy in action. The complete action

cluster with the high valued action along with the movement to the designated location is then instantiated in the course of action linkage. At this point the effects of random processes are examined.

Each random process model consists of a set of absolute and conditional functions that examine the course of action and check for applicability. Certain processes only affect certain actions, e. g., heavy traffic affects only movement. The process model functions produce a percentage probability of the event taking place, given the area, the store, and the time of day, which is then checked against the threshold of concern for this projection. If the threshold is exceeded then the appropriate action is taken to note the effects of this event on the planned course of action. This threshold of concern can be set to any level by the user of the Tactical Assistant.

After all the random processes have been applied, processing continues with the examination of the actions that have not yet been included in the course of action. Again the highest valued action is taken and run through the above described processing.

Eventually all the actions are linked into the course of action for this subspace. This subspace, its course of action, all the events posted by random processes and the level of concern used to select random events are now a projection of the hypothetical course of events.

The processing of this phase continues by examining the next subspace and performing the same construction and analysis on it as was described above. When all the projections have been produced the complete scenario consisting of the entire set of subspaces, linkages, notes, etc, is then passed to the next phase for final processing and presentation to the user.

C.2.4. Phase 4 - Presentation of the Scenario

The last phase of processing takes the complex structure created during the previous three phases and presents it in a simple form to the user.

The routines implemented at the present are not very clever or spectacular. There are routines to review the goals of a run, the primary courses of action developed, and the effects of random processes. The primary course of action is the one that was built using the tactical hierarchy of meet-appointments, fulfill highest-value goals, and minimization of travel. In addition routines were implemented to save such information in disk files for collection and preservation in hard copy form.

To see the complete set of alternative courses of action requires use of Grasper [LOWR77] print functions. The resulting display is complex to the naive user but since no such users were using this

version of the system it was decided not to pursue more accessible display and interaction routines at the present time. This lack of user oriented display routines would also make it difficult for the naive user to use the generated scenarios as an aid during "execution" of the plan.

C.3. Experiments Performed Using Simulation

Experiments were run on the Tactical Assistant to see if it could perform as expected. They included experiments with two, three, four and seven goals. Each experiment was run to produce a scenario that consisted of projections for each of five different thresholds of concern.

The thresholds represented different attitudes about the future. The first was set "unrealistically" high, producing an over optimistic "perfect" view of a future where nothing goes wrong. The second was the other extreme; set "unrealistically" low it produced an over pessimistic "Murphy's Law" view of a future where everything goes wrong. In between these two extremes were three "reasonable" settings that gave somewhat more moderate views of the future.

It was felt that a Tactical Assistant that produced a scenario consisting of these five projections could give the user a reasonable view of the possibilities. The two extreme views were presented first

to give an idea of the bounds of the "scenario space". The next three projections presented gave a "middle of the road" view. If the user were habitually overconfident or overcautious then the extreme views could be thought of as the most reasonable.

Experiment number one was the initial test of the system, its use of the planning heuristics, synchronization of events, use of thresholds and the effects of random processes, etc. It was done with two goals specified by the user. The system was able to handle this, showing the user the five different projections.

Experiment two was done with three goals, with the times of completion widely separated. One could see, from the five projections the possible effects of random events.

Experiments three through five involved the three goals due at the same time. The difference in the experiments was the degree of specificity of the goal due times. Number three had all three goals with the same mandatory due time. This forced each action into a different subspace. The user then had a choice: either only one of the goals could be accomplished or the due times/degree of specificity of some of the goals must be changed.

Experiment number four was a change in one of the goal time specificities to "suggested". The actions were now split into only two sub-spaces, each a choice of performing one of the conflicting actions plus the non-mandatory action.

The last of this series of experiments, number five, has only one mandatory goal time. Here all the actions were compatible and no splitting was necessary.

Experiment six was run with four goals. It had two sets of conflicting mandatory time goals, the first pair occurring early in the morning and the other later in the afternoon, avoiding any time conflict.

The scenario produced shows the breaking of the main group of actions into four subchoices - one bifurcation caused by the early time conflict and the other by the later time conflict.

The last experiment, number seven, was run with all seven possible tasks requested. The system had to make travel decisions, timing decisions, etc. The results show the system in its full working glory, as well as the problems caused by strict observance of the heuristics as they were stated. To get a "more reasonable" plan would require either a different ordering of the heuristics or else a larger and more clever means of applying them.


```
(get-tool
  (NAME (VALUE (get-tool))) ;name
  (DUE-DATE-OF-GOAL (VALUE ((d-sug 10)))) ;time due
  (VALUE-OF-GOAL (VALUE (9))) ;value
  (LOCATION-OF-GOAL (VALUE ((l-hdw x)))) ;where to go
  (TYPE-OF-GOAL-OBJECT (VALUE (i-tool))) ;what to get
  (TYPE-OF-GOAL-VERB (VALUE (v-get))) ;how to do it
  (TYPE-OF-GOAL (VALUE (a-transact)))) ;class
```

```
(go-to-movie
  (NAME (VALUE (go-to-movie))) ;name
  (DUE-DATE-OF-GOAL (VALUE ((d-mand 20)))) ;time due
  (VALUE-OF-GOAL (VALUE (8))) ;value
  (LOCATION-OF-GOAL (VALUE ((l-movies x)))) ;where to go
  (TYPE-OF-GOAL-OBJECT (VALUE (i-movie))) ;what to get
  (TYPE-OF-GOAL-VERB (VALUE (v-get))) ;how to do it
  (TYPE-OF-GOAL (VALUE (a-transact)))) ;goal class
)
```

```
(SEGMENT segmentxaaa)
(Random Threshold 10000) ;Set high so no random
                        ;events are considered
```

```
(PROJECTION projectionxaaa) ;The projection of the
                            ;future with this
                            ;assumption about the
                            ;effects of random
                            ;processes.
```

```
(ACTION first OCCURS AT s-home) ;All projections begin
                                ;at home
(TIME 1) ;The beginning of the day
(ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2);Travel
                                ;to shopping center 2
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2);purchase food
(TIME 8) ;purchase only takes one
                                ;time unit
(ACTION actionxaab:get-tool OCCURS AT s-ss2);purchase tool
(TIME 17) ;begin trip to mall
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls);travel
(TIME 20) ;Time movie begins
(ACTION actionxaac:go-to-movies OCCURS AT s-mls);Watch movie
```

```

(SEGMENT segmentxaab)
(Random Threshold 0) ;Lowest setting "everything goes wrong"

(PROJECTION projectionxaaa) ;Projection for the
                        ;"Murphy's Law"
                        ;random process assumption
(ACTION first OCCURS AT s-home); begin at home
(TIME 1)
(ACTION movexaab:goto-s-ss2 OCCURS AT s-ss2)
("action:" (movexaab) "may be delayed due to heavy traffic")
                        ;warning of effect of random process.
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
("action:" (get-groc) ("may fail to achieve goal.  Retry
                        should be made at location(s):" (s-ss1)))
                        ;This grocery may be out
                        ;of one of the items desired.
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
("action:" (get-tool) "may fail to achieve goal and no retry
                        is possible") ;There are no other hardware stores.
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
("action:" (movexaaa) "may be delayed due to heavy traffic")
(TIME 20)
(ACTION actionxaac:go-to-movies OCCURS AT s-mls)
("action:" (go-to-movie) "may fail to achieve goal and no
                        retry is possible") ;This movie is not showing anywhere
                        ;else in town

```

```

(SEGMENT segmentxaac)
(Random Threshold 75) ;an intermediate setting

(PROJECTION projectionxaaa);projection of a
                        ;"reasonable" assumption
                        ;about foreign processes.
(ACTION first OCCURS AT s-home)
(TIME 1)
(ACTION movexaad:goto-s-ss2 OCCURS AT s-ss2)
(TIME 7)
(ACTION actionxaaa:get-groc OCCURS AT s-ss2)
(TIME 8)
(ACTION actionxaab:get-tool OCCURS AT s-ss2)
(TIME 17)
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)

```

(TIME 20)
 (ACTION actionxaac:go-to-movies OCCURS AT s-mls)

(SEGMENT segmentxaad)
 (Random Threshold 50) ;Halfway between perfect
 ;and terrible.

(PROJECTION projectionxaaa) ;Another projection to outline
 ;intermediate possibilities

(ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION movexaaf:goto-s-ss2 OCCURS AT s-ss2)
 (TIME 7)
 (ACTION actionxaaa:get-groc OCCURS AT s-ss2)
 (TIME 8)
 (ACTION actionxaab:get-tool OCCURS AT s-ss2)
 (TIME 17)
 (ACTION movexaaa:goto-s-mls OCCURS AT s-mls)
 ("action:" (movexaaa) "may be delayed due to heavy
 traffic");Malls will sometimes become crowded
 ;near the time the movies start.

(TIME 20)
 (ACTION actionxaac:go-to-movies OCCURS AT s-mls)

(SEGMENT segmentxaae)
 (Random Threshold 25) ;Assume most things go wrong

(PROJECTION projectionxaaa)
 (ACTION first OCCURS AT s-home)
 (TIME 1)
 (ACTION movexaah:goto-s-ss2 OCCURS AT s-ss2)
 ("action:" (moveaah) "may be delayed due to heavy traffic")
 (TIME 7)
 (ACTION actionxaaa:get-groc OCCURS AT s-ss2)
 ("action:" (get-groc) "may fail to achieve goal due to lack
 of money".
 "please schedule trip to the bank and re-run".)
 ;Rather than the item being
 ;sold out, the groceries
 ;could cost too much.

(TIME 8)
 (ACTION actionxaab:get-tool OCCURS AT s-ss2)
 ;No trouble

```
                ;buying the tool.  
(TIME 17)  
(ACTION movexaaa:goto-s-mls OCCURS AT s-mls)  
("action:" (movexaag) "may be delayed due to heavy traffic")  
(TIME 20)  
(ACTION actionxaac:go-to-movie OCCURS AT s-mls)  
("action:" (go-to-movie) "may fail to achieve goal and no  
  retry is possible")
```

Figure 4: Output from the Simulation Based Tactical Assistant

D. By Retrieval and Modification of Experience

Once the Constrained Scenario Generation System was built it was possible to take a domain such as Planning Errand Running and construct routines to generate scenarios for it. The Constrained Scenario Generation System was the implementation of a domain independent Tactical Assistant, and is discussed in Chapter Five. It creates course of events examples by retrieving experiences of similar problems from a knowledge base and modifying them to fit the current problem exactly.

A simple knowledge base of model experiences showing possible courses of action given different tactical choices was easily built. The tactical choices were the tactics: least-travel, highest-value and meet-appointments. There was no heirarchy of tactical decision as in the simulation method. Only the pure tactic itself was used. Thus when pursuing the highest-value tactic attempting to meet-appointments was not considered. The experiences in the knowledge base also show

the effects of the random processes heavy-traffic, item-sold-out and insufficient-funds.

Figure 5 shows a simple example of a scenario generated by this method.

```

;the frame used to direct the recovery of similar
;experiences - those with goals similar to this.
goal frame: err-goal-2           ;the name of this frame
name: "22-FEB-198217:54:13.66"  ;the internal name
type: purchases                 ;type of goal
value: 5                        ;overall value
time: 0                          ;when to start
specifics: ((tool 9 (sug 10))
            (food 3 (sug 3))
            (movie 8 (mand 20))))
          ;form is: (item value (due date))

```

```

Segment: 1 Projection: 1        ;first segment,
                                ;first projection
Experience: x:err-g3:0007      ;the name of the
                                ;relevant experience
Goal: ((tool 9 (sug 10))
       (food 3 (sug 3))
       (movie 8 (mand 20)))
Tactic: meet-appts           ;the tactic this is an example of
Random Effects: helpful      ;assumption about the
                                ;random processes

```

```

Time Action
21: Drive to mls             ;Note order of travel
22: Purchase/Transact of: (movie) ;first errand
23: Drive to sc-2           ;Next stop
24: Purchase/Transact of: (tool food) ;handle two errands

```

Segment: 2 Projection: 1 ;Second segment, first projection
 Experience: x:err-g3:0008
 Goal:((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
 Tactic: meet-appts
 Random Effects: neutral

Time Action

21: Drive to mls ;parking is not a problem
 22: Purchase/Transact of: (movie) attempted, but (movie)
 sold out
 Will require a trip to one of ((mls)) in order to
 complete task ;Expected problems
 23: Drive to sc-2
 24: Purchase/Transact of: (tool food)

Segment: 3 Projection: 1
 Experience: x:err-g3:0009
 Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
 Tactic: meet-appts
 Random Effects: detrimental ;everything goes wrong

Time Action

21: Drive to mls delayed due to heavy traffic/trouble
 parking
 23: Purchase/Transact of: (movie) attempted, but (movie)
 sold out
 Will require a trip to one of ((mls)) in order to
 complete task
 24: Drive to sc-2 delayed due to heavy traffic/trouble
 parking
 26: Purchase/Transact of: (tool food) attempted, but
 (tool food) sold out
 Will require a trip to one of ((sc-2) (sc-1
 sc-2)) in order to complete task

Segment: 4 Projection: 1
Experience: x:err-g3:0001
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: highest-value
Random Effects: helpful ;things go correctly

Time Action

1: Drive to sc-2 ;Note change in order of travel
2: Purchase/Transact of: (tool food)
3: Drive to mls
4: Purchase/Transact of: (movie)

Segment: 5 Projection: 1
Experience: x:err-g3:0002
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: highest-value
Random Effects: neutral

Time Action

1: Drive to sc-2
2: Purchase/Transact of: (tool food)
3: Drive to mls delayed due to heavy traffic/trouble
 parking
5: Purchase/Transact of: (movie)

Segment: 6 Projection: 1
Experience: x:err-g3:0003
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: highest-value
Random Effects: detrimental

Time Action

- 1: Drive to sc-2 delayed due to heavy traffic/trouble parking
- 3: Purchase/Transact of: (tool food) attempted, but (tool food) sold out
Will require a trip to one of ((sc-2) (sc-1 sc-2)) in order to complete task
- 4: Drive to mls delayed due to heavy traffic/trouble parking
- 6: Purchase/Transact of: (movie) attempted, but (movie) sold out
Will require a trip to one of (mls)) in order to complete task

Segment: 7 Projection: 1
Experience: x:err-g3:0004
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: least-travel
Random Effects: helpful

Time Action

- 1: Drive to sc-2
- 2: Purchase/Transact of: (tool food)
- 3: Drive to mls
- 4: Purchase/Transact of: (movie)

Segment: 8 Projection: 1
Experience: x:err-g3:0005
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: least-travel
Random Effects: neutral

Time Action

- 1: Drive to sc-2
- 2: Purchase/Transact of: (tool food)
- 3: Drive to mls delayed due to heavy traffic/trouble parking
- 5: Purchase/Transact of: (movie)

Segment: 9 Projection: 1
Experience: x:err-g3:0006
Goal: ((tool 9 (sug 10))
 (food 3 (sug 3))
 (movie 8 (mand 20)))
Tactic: least-travel
Random Effects: detrimental

Time Action

- 1: Drive to sc-2 delayed due to heavy traffic/trouble parking
- 3: Purchase/Transact of: (tool food) attempted,
but (tool food) sold out
Will require a trip to one of ((sc-2) (sc-1
sc-2)) in order to complete task
- 4: Drive to mls delayed due to heavy traffic/trouble parking
- 6: Purchase/Transact of: (movie) attempted,
but (movie) sold out
Will require a trip to one of ((mls)) in order
to complete task

Figure 5: Output from the Retrieval And Modification Of Experience Based Tactical Assistant

The Retrieval and Modification of Experience version of generating scenarios was able to perform the same functions as the Simulation version, but the implemented random process models were not as detailed. In errand running, the situation analysis features of the Constrained Scenario Generation System were not used, since the situational parameters were not considered important enough.

The tactics are stored in the knowledge base as model experiences. Figure 6 shows an example experience.

```

experience frame: err-g3:0003
name: "21-FEB-198211:20:03.44"
type: model
tactic: highest-value
random-effects: detrimental
location: home
destination: (sc-1 sc-2 dtn mls)
goal: ((newspaper 5 (sug 10))
      (money 4 (sug 10))
      (food 3 (sug 10)))
coa: ((repeat (*traffic-thresh
              *soldout-thresh
              *insfunds-thresh)
             (setq *traffic-thresh
                   1
                   *soldout-thresh
                   2
                   *insfunds-thresh
                   ;random events classification thresholds
                   2)
      begin (move (car *min-locs))
            ;move to next required stop
            (if (greaterp:i (subset-position
                            ;check for traffic
                            *curloc
                            $$random-loc-classes)
                  *traffic-thresh)
                (heavy-traffic *curloc))
            (transact (intersection *items
                                   ;perform errands
                                   (choose *curloc
                                           $$items-offered
                                           car
                                           cadr)))
            (if (not (null
                     (items-above *soldout-thresh)))
                (sold-out ;check for sold out items
                          (items-above *soldout-thresh))
                (retry (items-above *soldout-thresh)))
            (setq *items (set-difference
                        ;remove from list of errands still to be done
                        *items

```

```
                                (set-difference
                                *curitems
                                (items-above
                                 *soldout-thresh))))
until (null (setq *min-locs (cdr *min-locs))))
```

Figure 6: An Example Model Experience

There are no true random process models as in the simulation method. Instead, each object of a random process (for example, each location of possible heavy parking) is given a rank, and for each assumption about random processes one rank level is selected and all items of that class or greater are then assumed to occur.

The discussion of the mechanics of retrieval and modification of experience are deferred to Chapter Five.

C H A P T E R V

THE USE OF SCENARIOS IN A COMPLEX DOMAIN

This chapter introduces the domain of Conflict Simulation Games and explains briefly why it is an applicable domain for the use of scenarios. The constrained scenario generation system that was built is described and compared to the original design. The highlights of a series of experiments that were performed with human subjects that examined the utility of the system are detailed. Some examples of what the system can do are presented. Finally, the knowledge base that was created is described and the system internal details are discussed.

A. Introduction to Conflict Simulation Gaming

A Conflict Simulation Game can be described as two or more forces operating in a common physical domain to achieve goals that are at cross purposes. Each force has units that may be manipulated to effect these goals. The problem for the intelligence behind a force is to decide how to organize and manipulate its units to make sure they accomplish the desired results.

The physical domain these games are concerned with is usually an abstraction of some actual historical conflict (e. g., the Battle Of Gettysburg), a "what-if" historical conflict (e. g., Operation Olympic - The invasion of the Japanese mainland in 1945), or a possible "what-might-be" contemporary conflict (e. g., the Warsaw Pact invasion of West Berlin in 1985).

The abstraction can be to a number of different levels leading to a number of different games (e. g., there are at least a dozen Battle of the Bulge games). The games are thus given complexity ratings from one (easiest) to nine (hardest). The abstraction can be along a number of different parameters, such as, spatial scale: a given distance on the game board represents how many actual kilometers; time scale: a game turn represents how many actual seconds, weeks or generations; or organizational scale: a force's unit may represent a single person, a company, an army.

In general a game has three main components: a game board on which the game is played, a set of counters that represent the units used by each force, and a set of rules describing how the game is played. The game board is normally a hexagonal array representing the actual territory of the conflict. Each element of this array is called a "hex". Each hex is then given a terrain type that is the abstraction of the type of terrain found in the actual territory, e. g., forrest, hills, city, desert, or river. The set of counters is

place on the game board and manipulated by the players within the rules of the game. The rules of the game describe the abilities and limitations of each player.

Conceptually the rules for a Conflict Simulation Game break into six sections:

1. A description (usually pictorial) of the game board - how terrain features are defined, where they are, what they mean, etc;
2. Rules about how the game is played, i. e., the Sequence of Play - when players can perform various forms of movement and combat;
3. Rules governing the movement of units on the board - usually in relation to terrain features and other units: which units can move through what type of terrain at what cost;
4. Rules governing combat between units - again usually in relation to terrain features and other units: what types of units can attack other units, strength ratios and possible results;
5. Rules describing the units involved and their abilities;
6. Rules describing the play variations - the victory conditions for each side, the length of the game, the initial positions of the units, etc.

B. Why Examine the Conflict Simulation Game Domain?

Conflict Simulation Game playing requires the bringing together of many diverse sources of knowledge, some sort of complex multi-level planning ability and above all, the power of reasoning.

In the following senses a conflict simulation game is much more complex than a conventional game such as Chess or Go:

- o The game board is much larger and has more positions that the game pieces can occupy.
- o Different places on the board mean different things (terrain and its effects).
- o More than one piece can move per turn. Usually any or all the pieces on a side can move during the course of a turn.
- o Each piece's movement is relatively unrestricted; i. e., each unit has a "movement allowance" with which it can move to any place on the board within the terrain restrictions.
- o Different units or differing unit types may have different movement abilities or ranges.

- o More than one combat (an attempt to disrupt or eliminate the opposing units) may occur per turn.
- o More than one unit may participate in a single combat (e. g., two friendly units could attack an enemy unit).
- o Units need not always be adjacent to engage in combat (e. g., artillery units).
- o Different units or differing unit types may have differing combat strengths.
- o A given unit may have one combat strength when attacking, and another when defending.
- o Combat is resolved by comparing the strengths of the units involved and cross-indexing with a random device such as a die.
- o Combat results do not always mean the elimination of units; results can vary from elimination to exchange of forces to partial damage to forced retreat to morale checks.
- o A single game may have a number of differing initial unit setups, reinforcement schedules, command and control limitations or victory conditions each based on a differing historical possibility resulting in completely different strategies and tactics (e. g., "what if Rommel had known exactly where the Allies

would land in Normandy?").

Thus even a relatively simple conflict simulation game such as the one considered in this work has a game tree with on the order of $10E250$ nodes after only ten turns versus a conventional game such as chess that has a around $10E120$ nodes after 50 moves [SHAN50], [deGR65].

The estimation of $10E250$ is derived from 40 units per side, each able to move through one hex, or two hexes, . . . , up to six hexes per turn; potentially any unit can combat any other unit in a turn, or two units could attack a single unit, . . . , up to six units attacking a single unit; different choices of who attacks who, etc.

B.1. Goals and Planning in the CSG Domain

There are a number of different sets or organizations of goals in this domain, e. g., hierarchical, negative, multiple and ranked. Hierarchical goals and planning consists in general of conceiving of the desired world state and the plan to achieve it at some abstract level and then in some manner increase the specificity until a detailed plan is developed. One goal or goal statement depends on the achievement of component subgoals. In Conflict Simulation Games the victory conditions for each force are given on a global game wide level leaving it up to the intelligence behind the force to devise

local detailed plans.

Negative goals are world states that are to be avoided. In Conflict Simulation Games they usually arise due to a desire to prevent some enemy force from achieving their goals.

Multiple goals are events distributed in time and space to be accomplished. In Conflict Simulation Games, multiple goals are sometimes stated explicitly in the victory conditions, or arise spontaneously from specification of abstract plans.

Ranked goals in the conflict domain are stated by the victory conditions. At the top, the best a side can achieve is a Decisive Victory. If a Decisive Victory is not possible the side may have to reorganize and replan to achieve a Substantial Victory. Worse yet would be merely a Marginal Victory. The scale continues with Draw, Marginal Defeat, Substantial Defeat or Decisive Defeat.

Some of the issues involved when processors are planning and executing in this domain include command and control, situation recognition, and tactical and strategic knowledge representation in addition to those of planning and problem solving.

A conflict simulation game is much more complex than a conventional game. Not only in just the raw counting of the number of different things that can happen in a single turn, but in the number

and kinds of knowledge necessary to play the game. For a computer system to begin to approach the game it must have some way of handling this knowledge.

A computer system working in the conflict simulation game domain must also have some way of dealing with inimical and random processes on a level above that of mere tree search. The complexity of these games precludes any attempt to use any tree search type analysis.

B.2. The Game Chickamauga

The game chosen for analysis in this thesis is Chickamauga [HARD75] published by Simulations Publications Incorporated of New York who have graciously let me use and duplicate their copyrighted material.

Chickamauga is a simulation on a Grand Tactical level of the battle which took place between the Union Army of the Cumberland and the Confederate Army of the Tennessee in September, 1863. The battle occurred around Chickamauga Creek, a tributary of the Tennessee River, in Northern Georgia. Historically, the South won the battle. Figure 7 shows a portion of the game board. The figure is a black and white translation of a full color image.

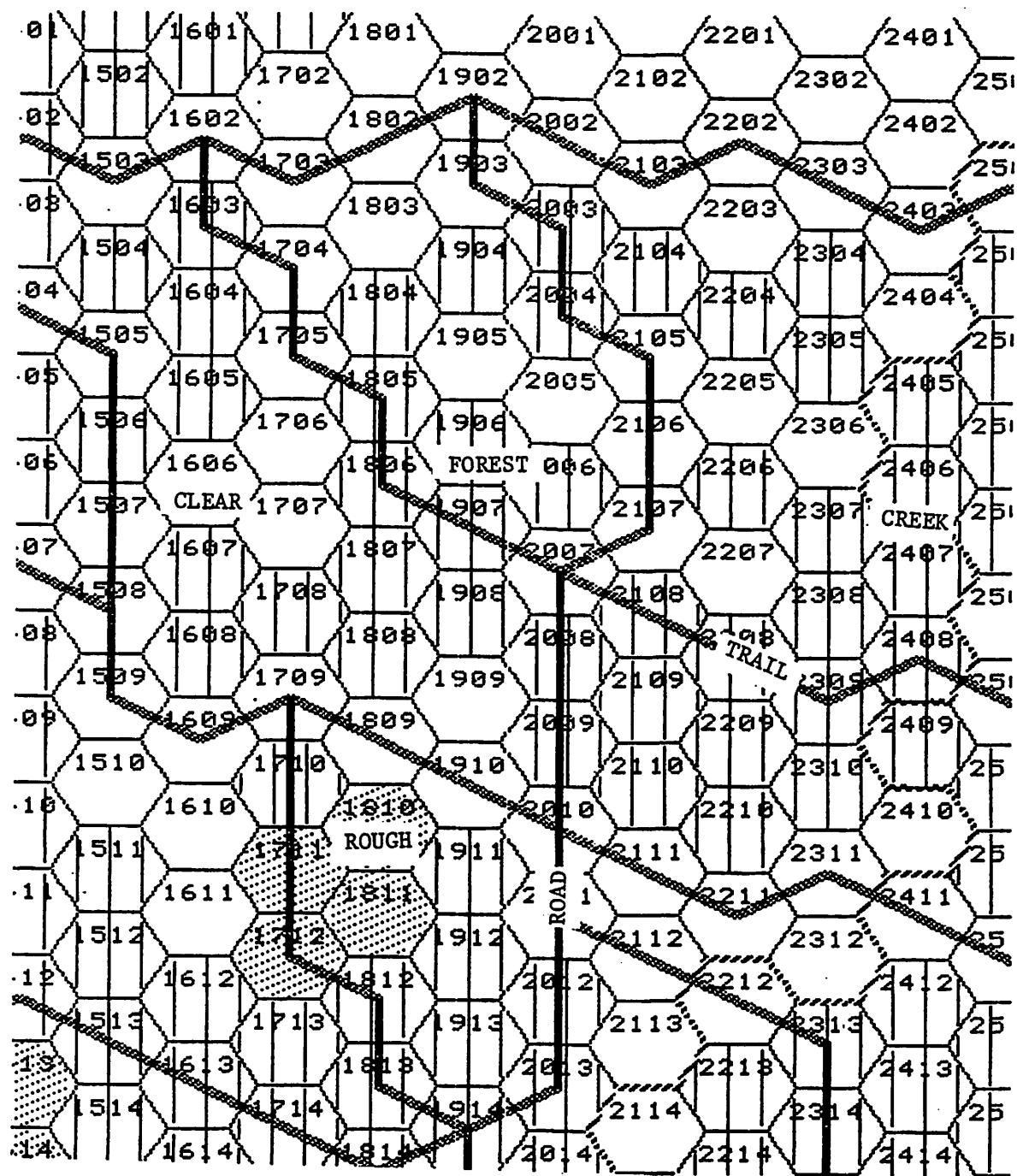


Figure 7. A Portion of the Mapsheet from Chickamauga.
Noted are the terrain types and features present.

Each side has about 40 units, some beginning the game on the board, some arriving later as reinforcements. Victory is determined by which player has the greatest number of victory points at the end of the game. Victory points are awarded, as play progresses, for the elimination of enemy combat strength points, and at the end of the game, for the occupation of certain hexes on the map. Points are also awarded for the exiting of units from the map thru specific exit hexes.

This game is considered to have a difficulty (for humans) of 4.3 on a scale of 1 (easiest) to 10 (hardest).

C. Using Scenarios to Solve the Problem

The features of scenarios that help solve this problem are categorical analysis, the generation of possible world examples by retrieval and modification of experience and their use in a tactical assistant.

C.1. Categorical Analysis

As mentioned above, the number of possible future states in this domain is combinatorially overwhelming. By the use of categorical analysis, this can be reduced to a manageable number of possible world categories. This is done by the five step process outlined in Chapter

Three, above. The first two steps involve choosing the domain parameters and their values with which possible world categories will be determined. Three parameters were chosen to delineate the conflict simulation game domain.

The first parameter is enemy-behavior. It is an attempt to capture the concern and interest the enemy player has in the area in question. It can take on two values: passive and aggressive. The first represents the assumptions that the enemy does not have an active goal to be accomplished in the area, and that its tactics will most likely be defensive. The second is the other alternative: the enemy has goals to be achieved and will act offensively to accomplish them.

The second parameter is enemy-tactic. It attempts to capture the actual kinds of course of action the enemy would pursue. It is not independent from the enemy-behavior parameter. Its values are the names of the courses of action that could be undertaken. For aggressive behavior the values could be frontal-attack, flank-attack, encirclement, or column-advance. For passive behavior the values could be line-defense, fluid-defense, or sit.

The last parameter is random-effects. It represents the class of results expected from combat. It also attempts to capture the idea of the actual completion of a given tactical action, which may be

dependent on terrain conditions that were abstracted out of the situational analysis. A value of helpful means the die roll on the combat results table would be the best possible and that the tactic would be completed successfully. Detrimental would select the possible worlds where things went badly. Neutral would be the worlds in between.

The third step of categorical analysis is that of choosing the possible courses of action that could be undertaken to achieve the goal-set in the current situation. Tactics for this domain are generated by a tactical analysis routine, built under the direction of a domain expert who provides abstract rules for the selection of course of action.

The fourth step of categorical analysis is that of selecting possible world categories in which to examine the courses of action. In the implemented system this is done by the constraint generation routines that make up constraints with which to query the knowledge base for relevant experiences. It uses knowledge provided by the domain expert, first to select which modes of enemy behavior are applicable (the only time both behavior modes are not included is if the situation does not have any enemy present). Second, the tactical analysis routine is called from the enemy's point of view to suggest tactics they might pursue. Finally, all three random effects classes are always considered.

The last step of categorical analysis is the generation of course of events examples to show what the possible futures are. This is covered in detail in the next section.

C.2. Retrieval and Modification of Experience

The examples of each course of events, in each possible world category, are generated by retrieval and modification of experience. This approach was chosen due to the complexity of the domain. An expert is given the task of selecting all the experiences that relate to the class of problems that the system will be asked to solve. These experiences are then placed in the knowledge base.

Retrieval means the selection, from the knowledge base of experiences, of a correct and complete set of relevant experiences to "cover" the possible futures. It is a complex problem discussed below in more detail. Once the situational and tactical analyses have been completed, the constraints they indicate, together with different choices for the values of the above listed parameters, comprise the queries to present to the experience base.

Modification refers to the fact that the experiences in the knowledge base will not fit the current problem exactly and will have to be modified. In some cases this may be simply instantiation - replacing slot values; in others it may be quite complex, requiring

the invocation and execution of functional elements attached to the experience.

Given that this selection process is done correctly, and the relevant experiences are in the knowledge base, using retrieval and modification of experience statements can to be made about possible futures after a search of the knowledge base. This search is bound by the size of the knowledge base rather than the size of the operator space, which in the conflict simulation game domain is effectively unbounded.

C.3. Tactical Assistant

The Tactical Assistant for this domain performs steps three through five of categorical analysis as described in Chapter Three. In addition, it must be able to present the resultant scenario to the planner in a meaningful form. A graphical display was chosen that would display the map sheet, the forces involved, and their activities.

By presenting the hypothetical courses of events in graphical form, the user is made aware of the potentials and possibilities easier than by non-graphical means. The Tactical Assistant also contains an access system that allows the user to easily peruse the various possible world categories.

D. Overview of the Constrained Scenario Generation System

A tactical problem consisting of a set of goals to be accomplished and a situation (the arrangement of forces) in which to accomplish them is presented to the system. The system returns an analysis of the possible futures: a course of events example for each of the possible courses of action that can be undertaken to achieve the goal, for each possible world category. This analysis, or scenario, is then presented to the user in graphical form.

At present the goal can be any set of requests for designated friendly forces to take or hold territory, or destroy enemy forces or protect friendly forces, expressed as a disjunction of conjuncts. The situation can be any arrangement of friendly and enemy forces. The situation analysis routine generates a description for each force involved in the problem. For example, if unit Gist (strength 5) is within 6 hexes of the road junction to be taken, its descriptor would be "(near gist 5)". The descriptors for all the forces are collected in a list that becomes the constraint expression for this facet.

The analysis is performed by retrieval and modification of experience as follows. A series of queries is constructed. The queries are presented to the Constrained Example Generation (CEG) [RISS82] system that acts as a database retrieval system to retrieve experiences of similar goals and situations.

The experiences in the knowledge base are made up of (1) a description of a previous problem, the goal and situation; (2) the choice of course of action that was made; (3) an "after the fact" description of the world category, e. g., if the random processes went badly then the random effects parameter would be "detrimental"; and (4) a description of the course of events as they actually occurred. The exact form of the experience is discussed in more detail below.

The results of the queries are then modified or instantiated to fit the exact current goal and situation. These modified experiences are now projections of hypothetical courses of action and are presented to the user, with the understanding that these experiences are an attempt to outline the range of possible futures without necessarily giving the exact probability of their hapenning again.

D.1. Constraints and Retrieval

Each query is a set of constraints. Each constraint is a list of two parts: the name of the facet that this constraint is concerned with, and the constraint expression - the value of the facet the selected data base item is to have. The exactness of the match between the constraint and data item is determined by a domain specific judgement routine (one for each facet). This judgement routine performs whatever syntactic and semantic manipulation is necessary to determine if the data item fulfills the constraint.

There are two constraints designed to select experiences that deal with a similar problem, and four constraints derived from steps three and four of the categorical analysis performed by the Tactical Assistant:

1. the type of goal (e. g., conjunction of acquisition or enemy force destruction subgoals).
2. the strength and disposition of the forces involved, as strength ratios (friendly/enemy) and semantic descriptors (e. g., "force one near the goal"). These first two constraints describe the problem the retrieved experience should concern.
3. the tactic to be used; this is from the tactical analysis routine,
4. the enemy behavior,
5. the possible enemy response tactic, and
6. the type of random effects to consider. These last three constraints select a possible world category.

The first two constraints are gotten straightforwardly from the goal as determined by the planner and the description of the situation by the situation analysis routine. The features of the goal/situation include the structure of the goal expression, the relationships of the

forces involved to the goals, the strength ratios between the forces involved, and the distances between them.

The values for the tactical constraints are chosen by a tactical analysis routine that compares number of forces on each side and suggests possible tactics (e. g., if a side has only a single unit then the encirclement tactic would not be proposed, since it requires at least two units to perform). The behavior and random effects constraints are selected so that a query is made for each possible combination. The set of queries thus contains a query for an experience with a similar goal and situation for each possible combination of course of action, enemy behavior, enemy tactic, and random effect.

The constraints used in the queries of the knowledge base performed by the CEG system are used by the judgement routines to compare the salient points of the tactical problem to the retrieved experiences. Because the CEG system already had mechanisms for examination and selection of items from a database on the basis of arbitrarily complex semantic features it was worthwhile to keep to the CEG format to take advantage of this facility.

D.2. Modification

Since a retrieved experience will rarely match the tactical problem under consideration it must be instantiated or in the case of model experience, modified. The retrieved experience is a "memory" of what happened with a similar problem in the past. Exactly what was meant by "similar" was decided by the domain expert who wrote the judgement routines that used the constraints to select this experience.

After the relevant experiences have been selected they are made to fit the current situation exactly by an instantiation routine, which fills in such features as the exact location and strength of the forces; if the relevant experience is a model experience, then the modification of the course of events slot is also necessary.

The scenario is then presented to the user by displaying the map including the goals and forces, and then drawing each movement and combat of the hypothetical course of events. There is a simple to use interface to help the user select the possible world assumption set to be displayed. For example, the user can ask to see all helpful worlds, or all worlds in which the enemy behaves aggressively and the random effects are detrimental.

Since the suggestions for course of action proposed by the tactical analysis routine and the enemy course of action proposed by the enemy-behavior and tactics are based only on abstract considerations without exact or detailed analysis of the goal and situation it is possible that not all queries will have answers. The answers in the knowledge base are all those that have by experience been deemed applicable. Those queries that were deemed inapplicable by the knowledge base are saved and included in the scenario, available for review by the curious user.

To allow the user to understand some of the complexity of the scenario being presented the graphics display system was developed. Although the graphics display gives a much better "feel" of the tactical flow of the problem to the gaming novice, it hides much of what would "really" be going on. In particular, the effects of terrain were ignored in the presentation to the user. It was felt that the benefits of comprehension due to this abstraction outweighed the cost of loss of detail.

The current system is capable of handling all tactical problems to a useful level of detail short of serious strategic considerations within the set framework of abstraction.

E. Human Study

Estimations of the utility of the Tactical Assistant are based on informal comments and reactions of novices and experts who were given a small test that determined the systems's ability to help them.

Transcripts were collected from interviews with the eleven subjects. The subjects were given four tactical problems and asked to describe how they would solve them. They were then shown the analysis of the same problems done by the Tactical Assistant. Finally, they were again asked to solve the problems. The difference, if any, in performance was measured and any improvement noted.

During each problem solving session the subjects were encouraged to discuss their own views of the future - what they thought the choices were, the results of courses of action, the possible enemy actions and final consequences.

E.1. Classes of Subjects

The subjects were divided into the following classes:

- A. Complete Novice - people who had never played any sort of tactical conflict simulation game. There were three complete novices tested.

- B. Inexperienced gamer - people who have seen or briefly played some sort of tactical conflict simulation game. There were three inexperienced gamers involved in the study.
- C. Gamer - people who have played tactical conflict simulation games but not Chickamauga. There were three gamers who participated.
- D. Expert - people who have played conflict simulation games including Chickamauga and are familiar with its tactics and strategies. There were two experts.

Although it was possible to classify the subjects by their experience it did not always follow that the experts gave the best answers and the novices the worst.

E.2. The Test Problems Given to the Subjects

The four problems presented to the subjects are shown in figures 8, 9, 10 and 11. Figure 8 shows the first simple problem. The goal of the two Confederate forces is to destroy the Union force. Figure 9 shows the second problem. The goal of the two Confederate forces is to destroy both Union forces.

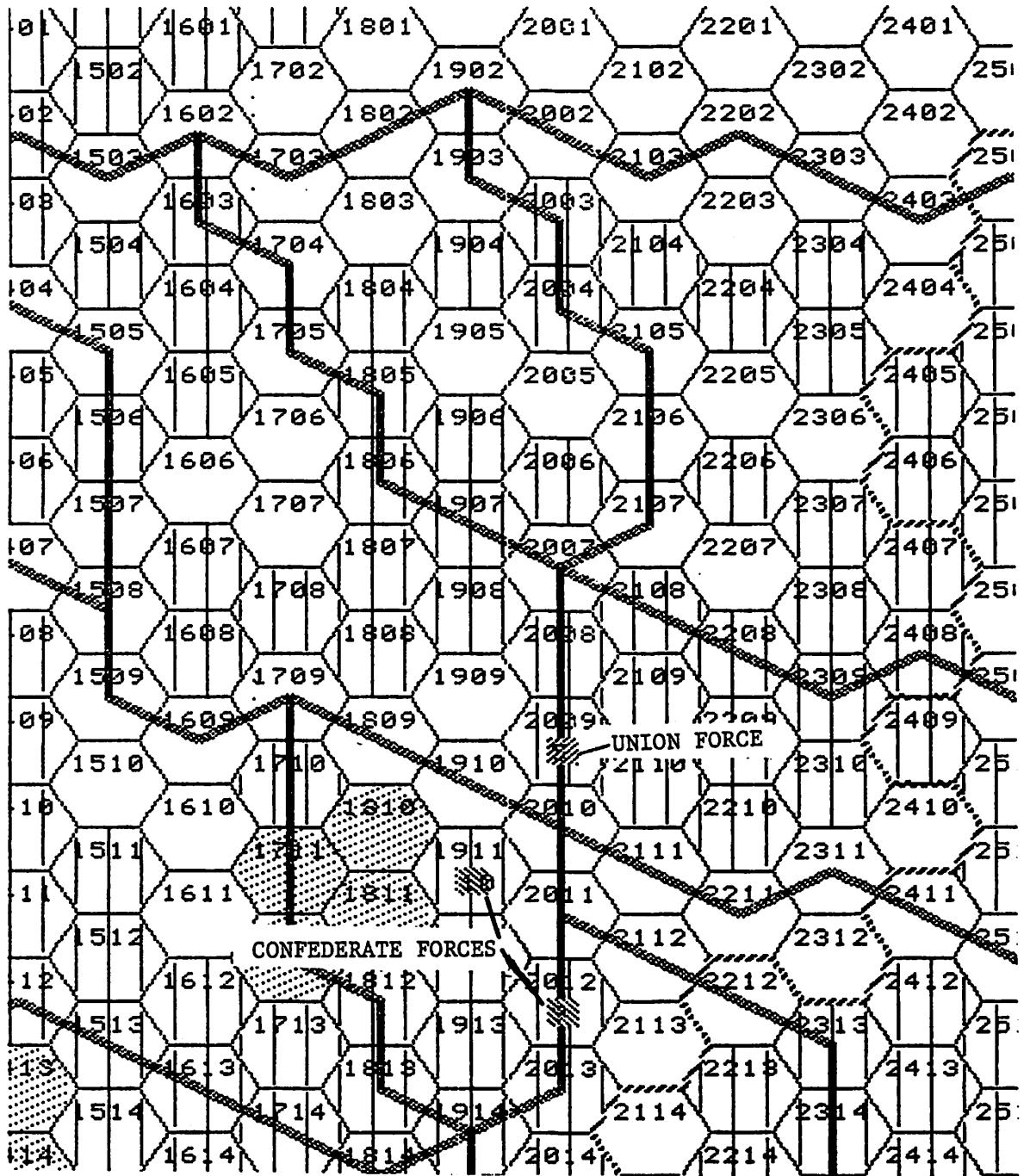


Figure 8. Problem 1. The two Confederate forces of strengths 10 and 9 must eliminate (remove from play) the Union force of strength 5.

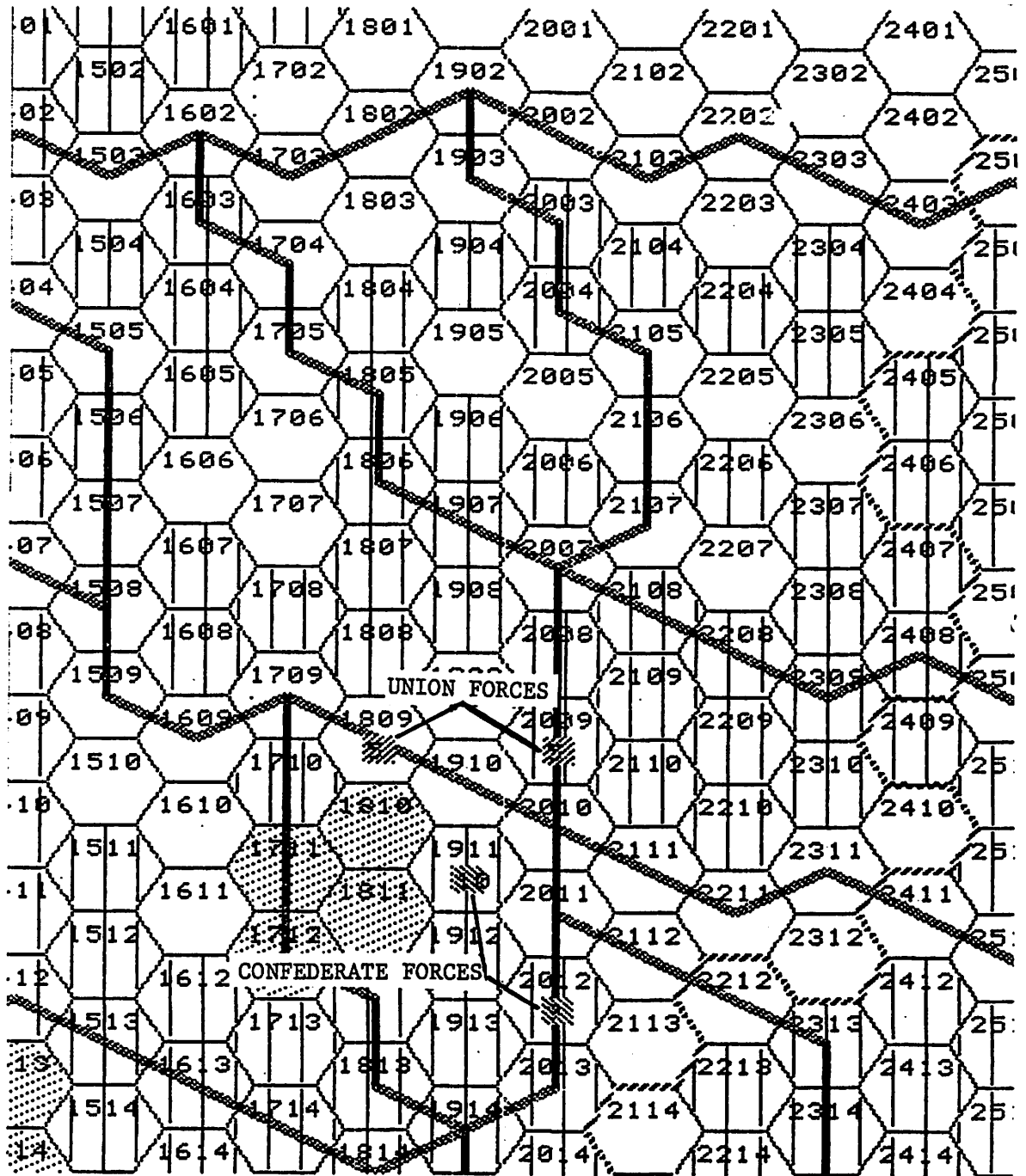


Figure 9. Problem 2. The two Confederate forces of strengths 10 and 9 must eliminate the two Union forces (each of strength 5).

Figure 10 shows problem three which involves a conflict of resources. The single Confederate unit must protect the road (highlighted by the series of road junctions) and itself from the two Union forces. The resources are the unit itself and the stretch of road. The conflict comes from the choice of protecting the unit by giving up territory or try to defend the territory by sacrificing the unit.

Figure 11 shows the final problem. The two Confederate forces must protect both highlighted road junctures plus themselves from the two Union forces.

E.3. Suggested "Correct" Solutions

For problem 1, the best solution is to try to encircle the Union unit to prevent its retreat. A purely frontal attack could take much longer to complete.

For problem 2, the Confederates should concentrate their forces on one Union unit hoping to destroy it quickly before the other Union unit gets away. If the forces are split to attack both Union units simultaneously, the odds will only be 1:1 and 2:1, perhaps allowing both to escape or even a successful counterattack.

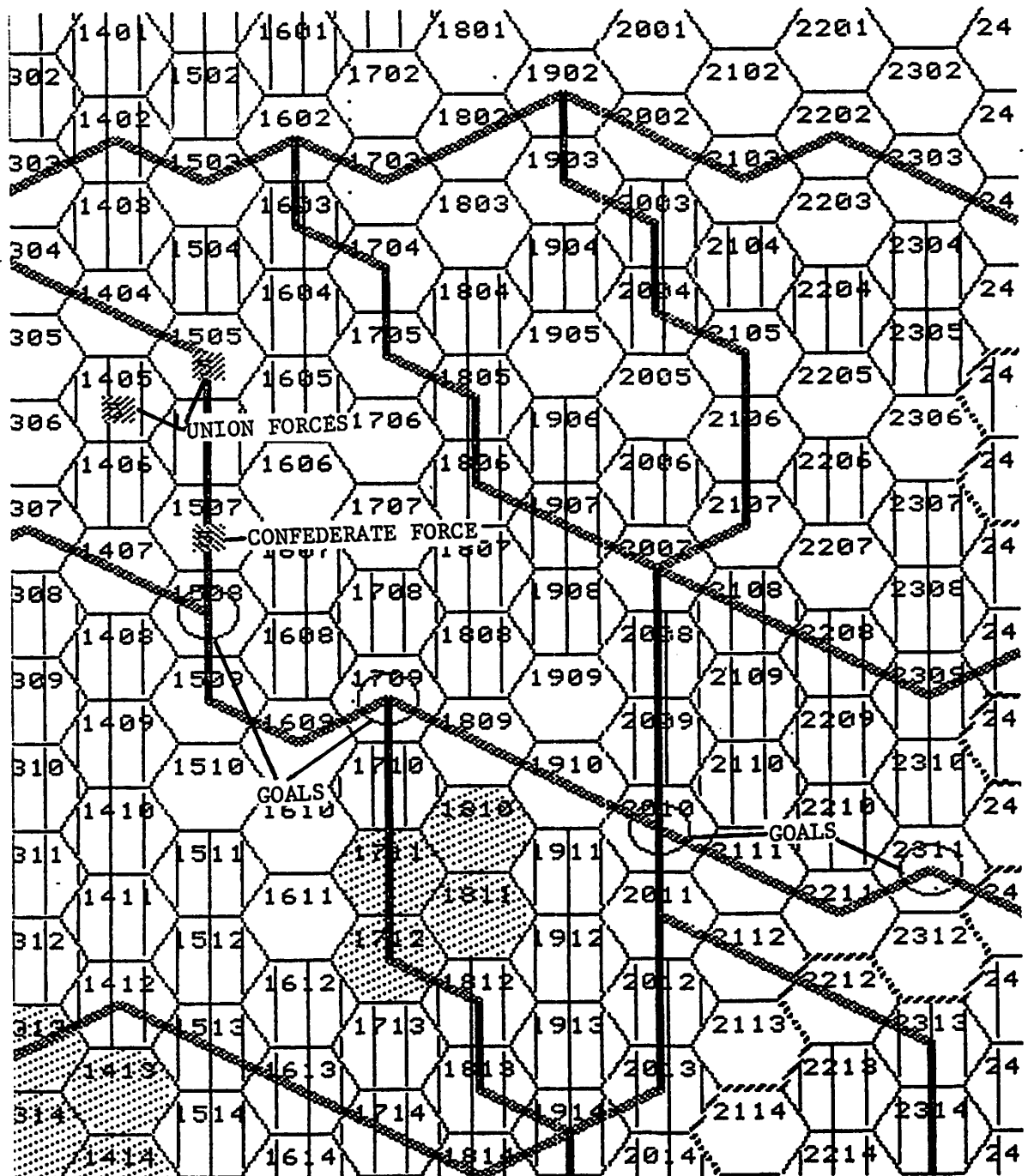


Figure 10. Problem 3. The Confederate force of strength 5 must protect itself from elimination and keep the two Union forces (each of strength 5) from occupying any of the road junctions leading to the bridge over the creek (black circles).

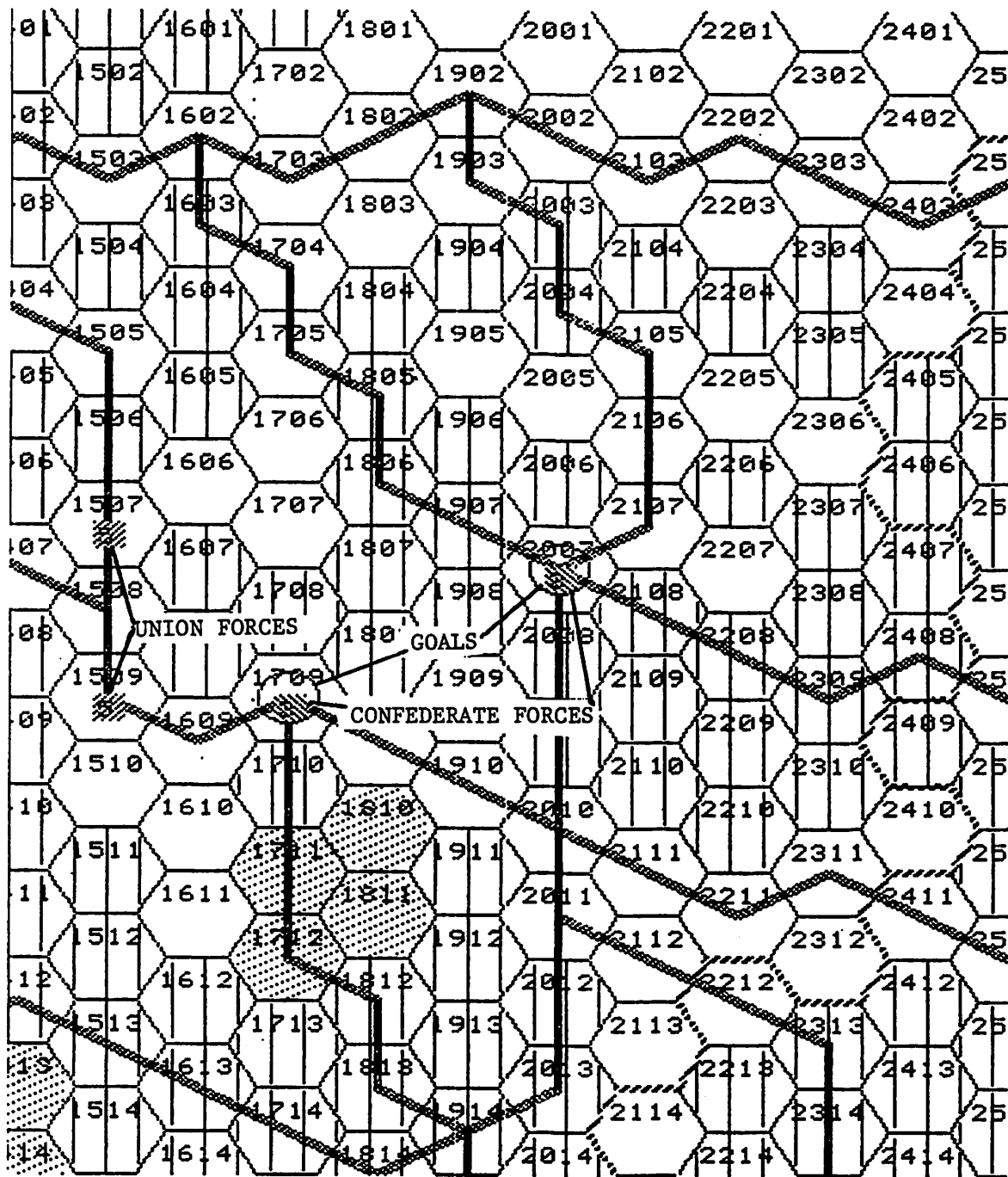


Figure 11. Problem 4. The two Confederate forces (each of strength 3) must protect themselves from elimination and keep the two Union forces (each of strength 5) from occupying either road junction (black circles).

Problem 3 is more difficult. It is hard to choose a "correct" course of action since there is no way to prevent the Union forces eventually taking the road with or without destroying the defender. All that can be chosen is how long it takes and where (if at all) the Confederate force is going to stand and fight (and perhaps be eliminated).

Problem 4 has a solution that is better than most of the other solutions. It is to use both units to form a line in defence of the leftmost goal. This solution is not perfect. The Union forces can attempt to flank and eliminate the units one at a time or sneak by to take the unguarded junction, but they cannot encircle the Confederates as they could if both Confederates were placed in the same hex or left to guard their road junctions alone.

E.4. Comments and Reactions of the Subjects

From the comments and reactions of the subjects it seemed clear that those of classes A and B were the most helped by the system while those of classes C and D were helped the least.

Class A subjects (the complete novices) found first that there were choices of course of action that they had not thought of. The class B subjects (inexperienced gamers) could think of most of the possible courses of action but ususally not all the possible outcomes

of each choice of course of action. After seeing the display they all chose the "more correct" solutions.

The class C and D subjects (gamers and experts) sometimes also did not think of all the courses of action or outcomes, but this was because they usually had a favorite course of action for a given situation type into which they tried to fit the problem. The alternative courses of action and outcomes were useful to see, but they would not have made those choices anyway. Some of the gamers got bogged down in the details of plotting the exact course of action and had trouble suggesting higher level abstract solutions even after being asked for such abstract solutions.

All of the subjects responded positively to the color graphics display of the scenarios. All felt that regardless of whatever detail was being abstracted out it was more meaningful than a non-graphics presentation would be.

Some of the novices mentioned that not having to try to understand or remember the combat results table when using the system was an aid.

One of the gamers mentioned that they needed a system that they could ask specific questions and get specific answers - for example, in problem 3 the question was "where can I put this guy [the Confederate unit] so he will be safe?"

One problem some of the novices had was inferring the "correct" solution from the sometimes large number of alternative futures presented. They would have liked the system to make suggestions or perhaps give explicit advice rather than merely presenting the possibilities.

Another problem was the lack of strategic information. This was most important to the solution of problem 3. They wanted to know what else was going on nearby. One novice (obviously a natural tactician) immediately recognized that the resources were insufficient to meet the goals. She wanted to know where her reinforcements were and when would they arrive?

Additionally, there was the question of what was the strategic implication of the tactical solution. The system could show the details of this problem, but what did that mean to the overall game situation? Is the player now in a better or worse situation than before?

In summary then, it appears that at this level of problem the gamers and experts could do as well without having the system available, but for the novices and inexperienced gamers a system like this would be of benefit. The problems that were mentioned were in some cases things that had been deliberately left out of the system (such as giving explicit advice of what to do), and in others, left

out due to the complexity of solving them (such as inferring strategic implications).

F. Examples of the System

This section shows an example of retrieval and modification of experience by showing one of the experiences retrieved as an example of one of the possible worlds in problem 1. It also shows some of the displays shown to the users trying to solve some of the other problems.

Figure 12 shows the course of action of one of the relevant experiences for problem 1. Figure 13 shows the same course of action after modification.

Figures 14 and 15 show the course of events of a modified experience for problem 2. The closest Union unit is attacked and destroyed, the second attempts to counterattack and is destroyed.

Figures 16 and 17 show an alternative outcome. Note the change in assumption about random effects. It begins the same but the Union unit is able to retreat instead of being eliminated. Together the two Union units successfully counterattack the weaker Confederate unit.

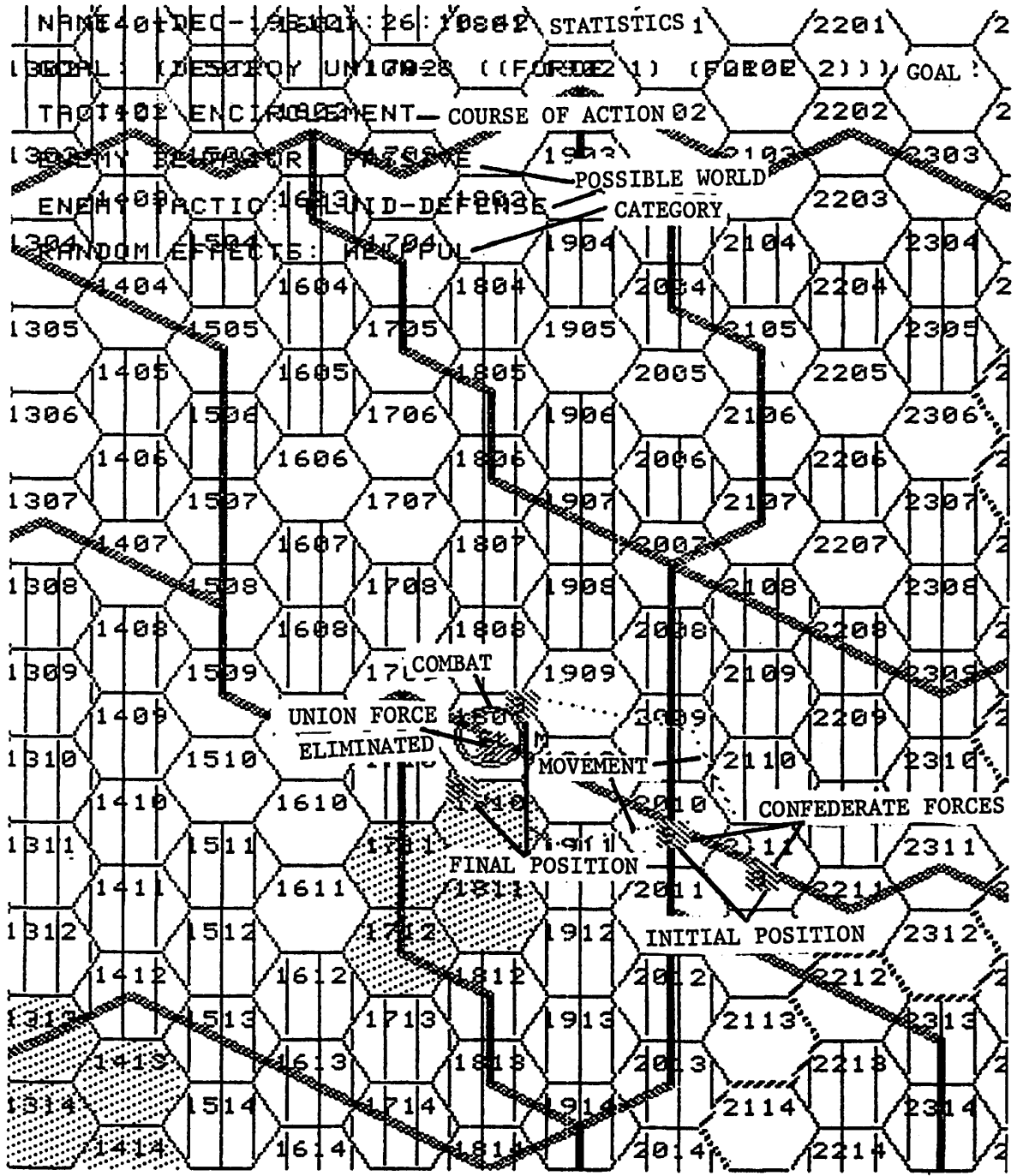


Figure 12. Relevant Experience from Problem 1. The experience is of two Confederate forces of strengths 8 and 9 moving to surround and eliminate a Union force of strength 5.

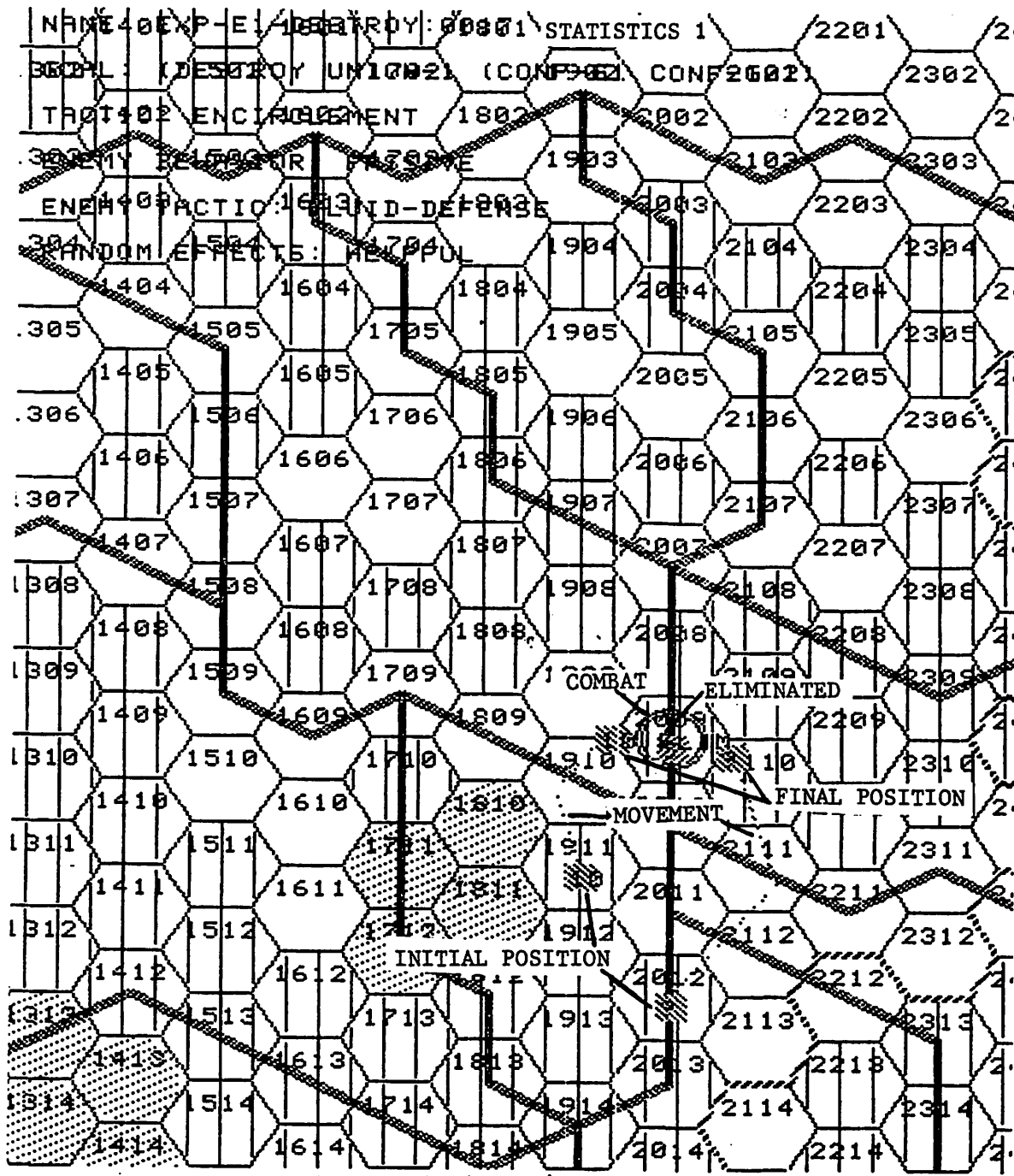


Figure 13. The Modified Experience. The experience from figure 11 has been modified to fit the actual situation of problem 1. The course of events stays the same.

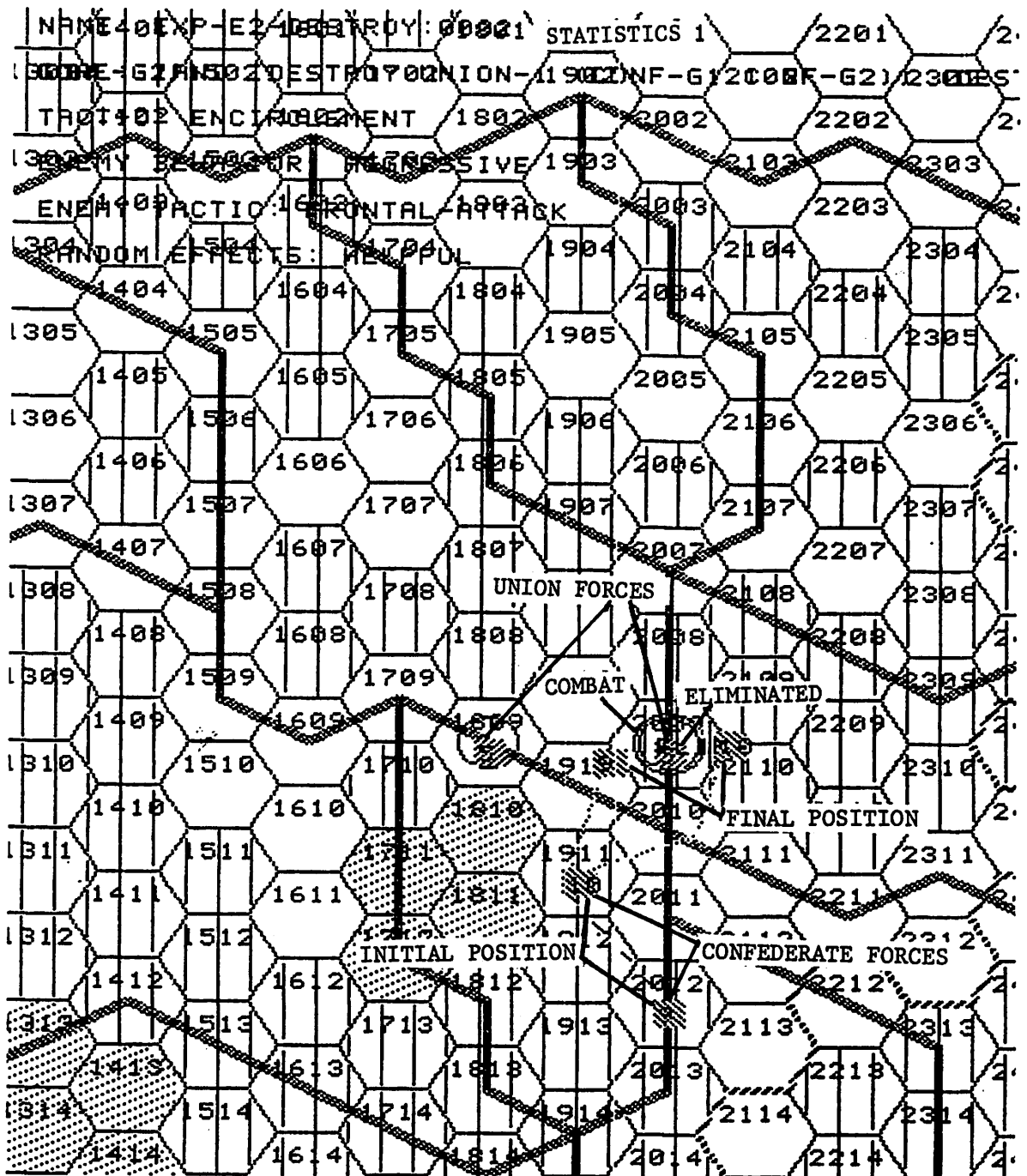


Figure 14. A Modified Experience from Problem 2 (turn 1).
 The Confederate forces surround and eliminate the first Union force.

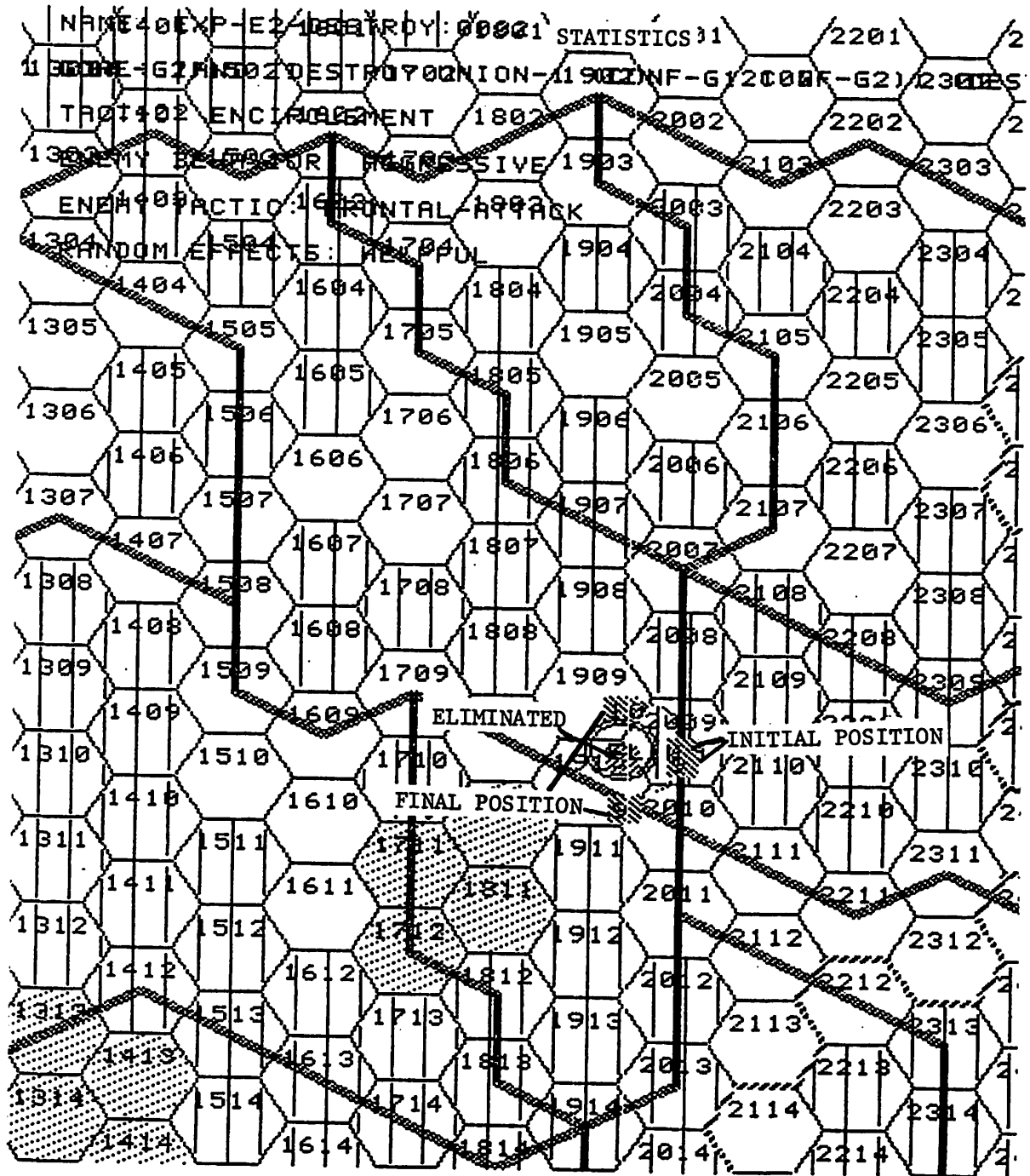


Figure 15. A Modified Experience from Problem 2 (turn 3). The remaining Union force tried to counterattack, but was then surrounded and eliminated.

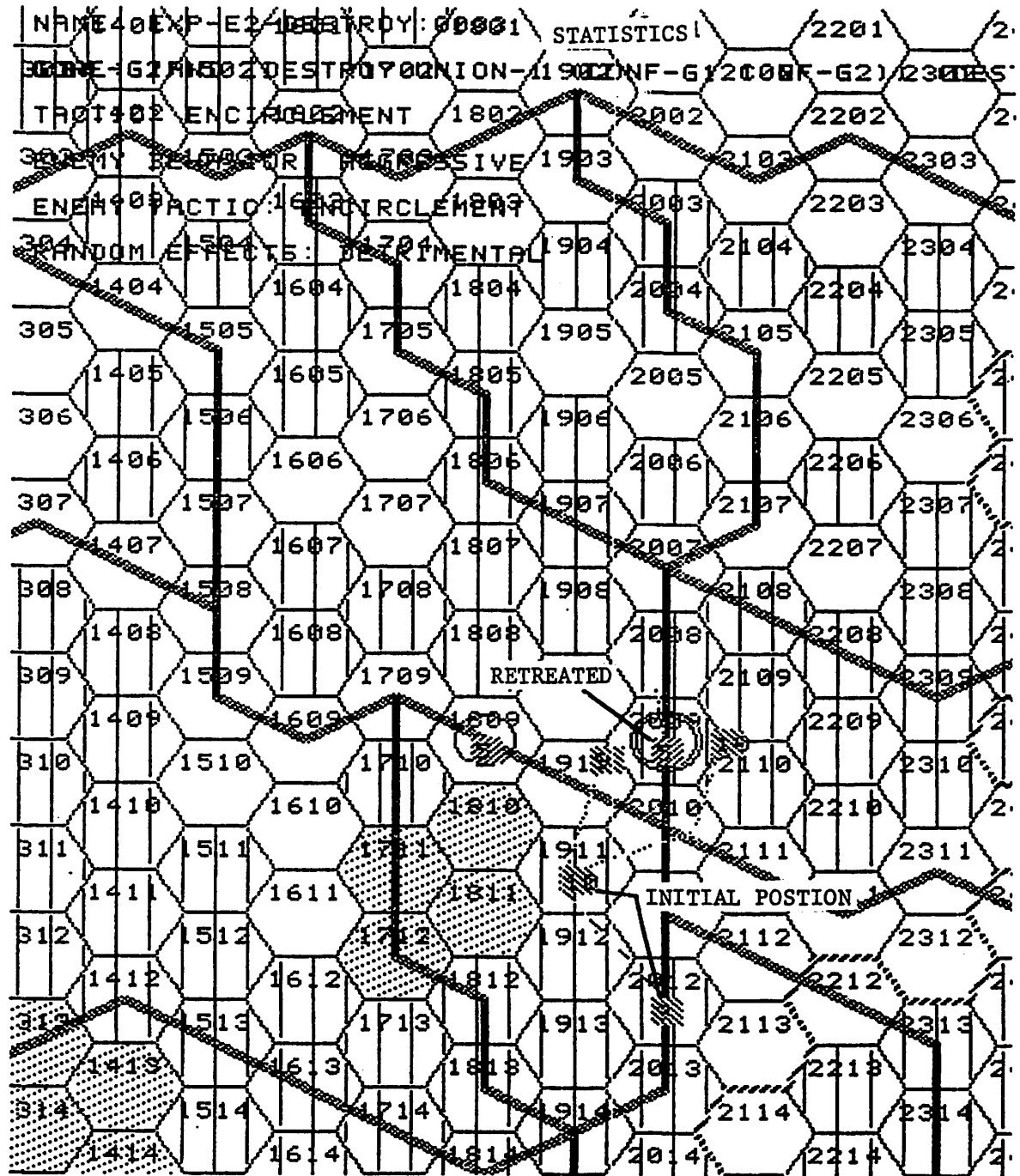


Figure 16. An Alternative Possible World Category from Problem 2 (turn 1). The Confederates try to eliminate the Union force as before, but only achieve a retreat result instead of elimination.

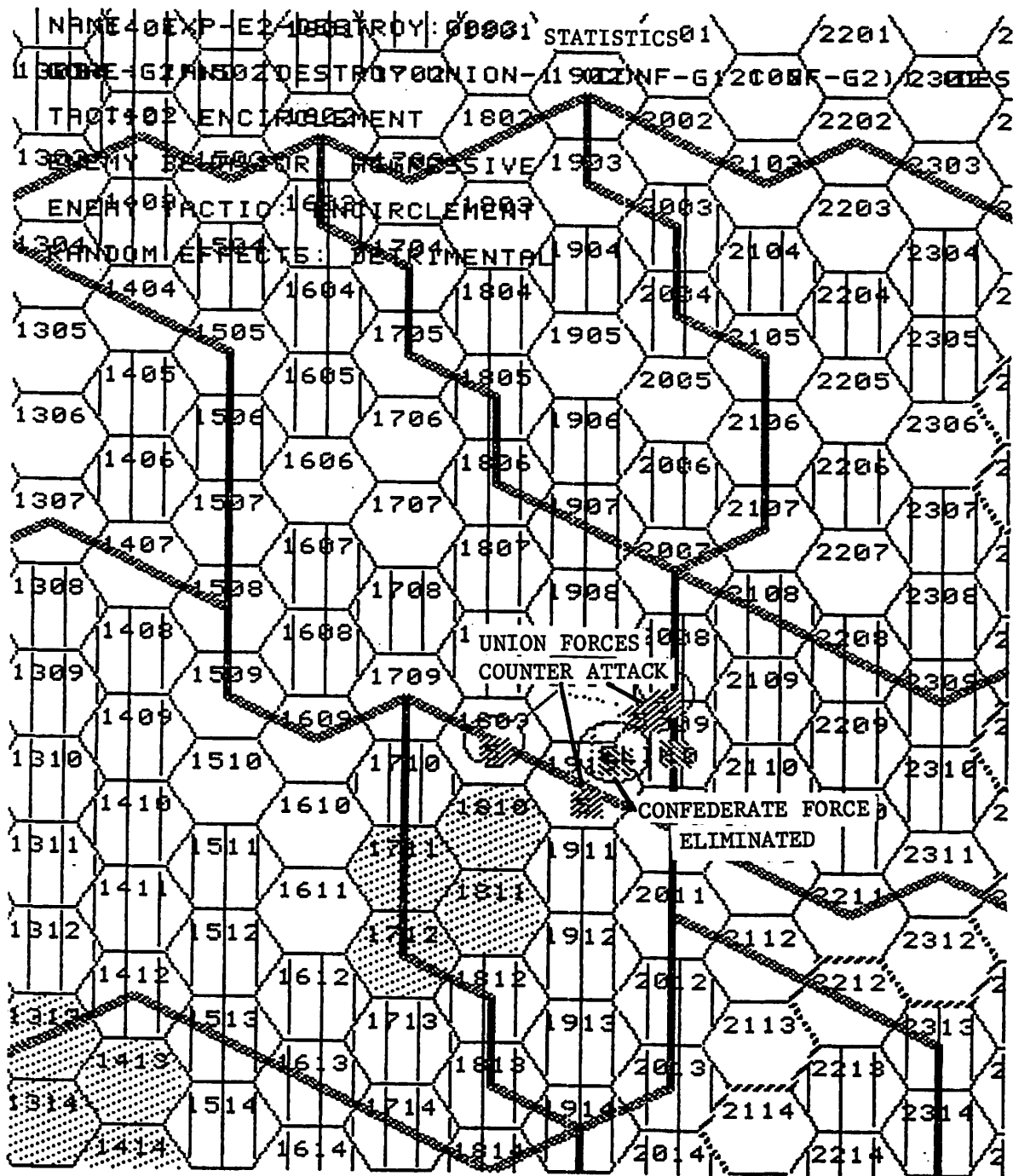


Figure 17. An Alternative Possible World Category from Problem 2 (turn 2). The Union forces counterattack the weaker Confederate force and eliminate it.

Figure 18 shows one of the experiences from problem 3. First the Confederate unit retreated to the first road junction. Then the Union units attack and are repulsed (note the random effects - helpful).

Figures 19 and 20 show an alternative - the Confederate unit is forced to retreat to the second junction, is pushed back again and again (not shown), eventually standing in front of the creek where it is surrounded and eliminated. Figure 21 shows another alternative - the Confederate unit attempts to stand and defend the first junction, where it is surrounded and destroyed.

Figure 22 shows one projection from problem 4. The two Confederate units were combined on the first junction. They were forced to retreat and the Union forces take the other unprotected junction. Figure 23 shows an alternative. The forces were kept on their respective goal junctions. One was surrounded and destroyed and the other was forced off its goal.

Figures 24 and 25 show a course of action alternative. The units formed a line to defend the first goal junction, the Union forces attacked but were driven back.

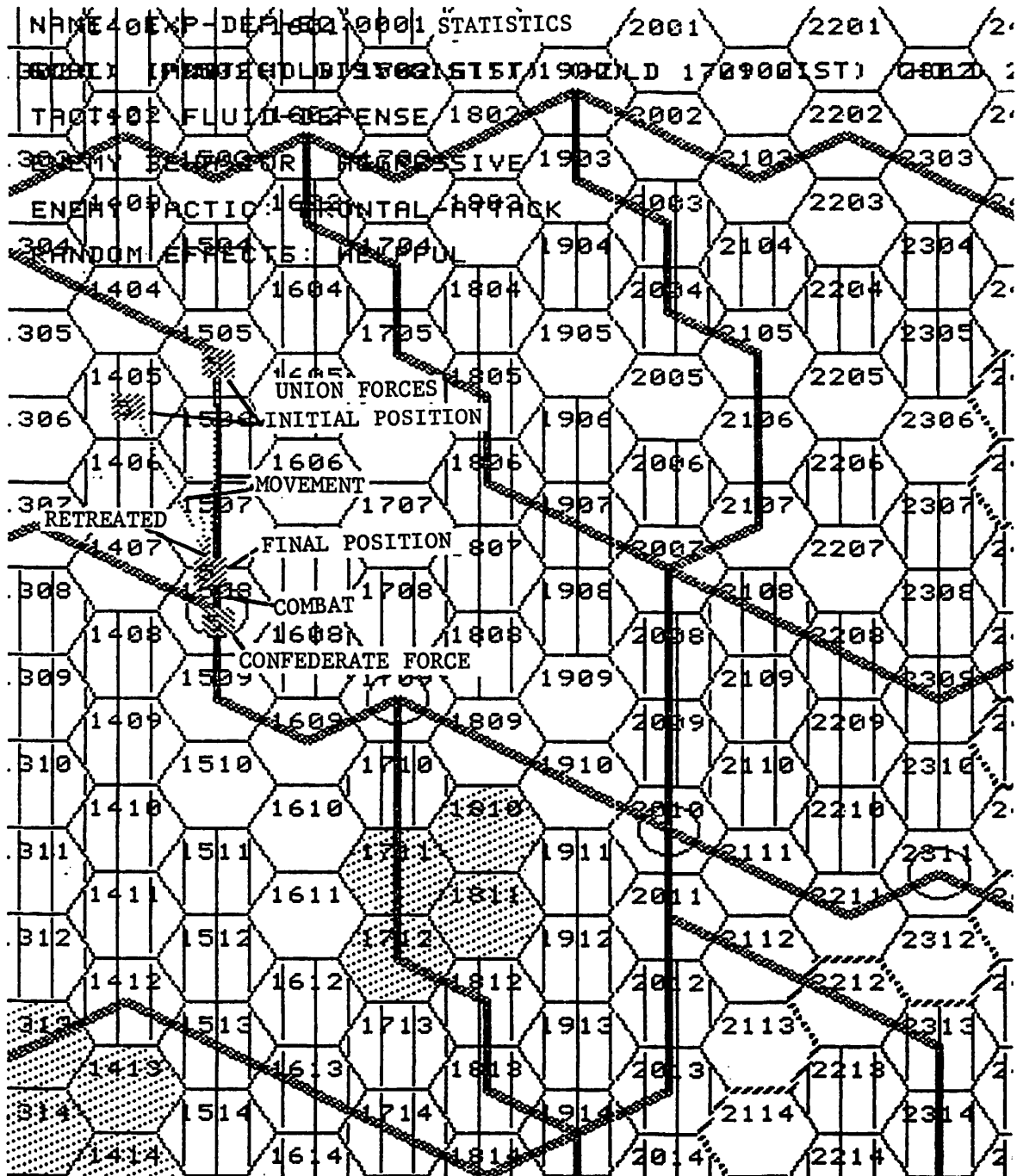


Figure 18. A Helpful Possible World Category from Problem 3. Every thing goes perfectly - the Union forces attack but are driven back.

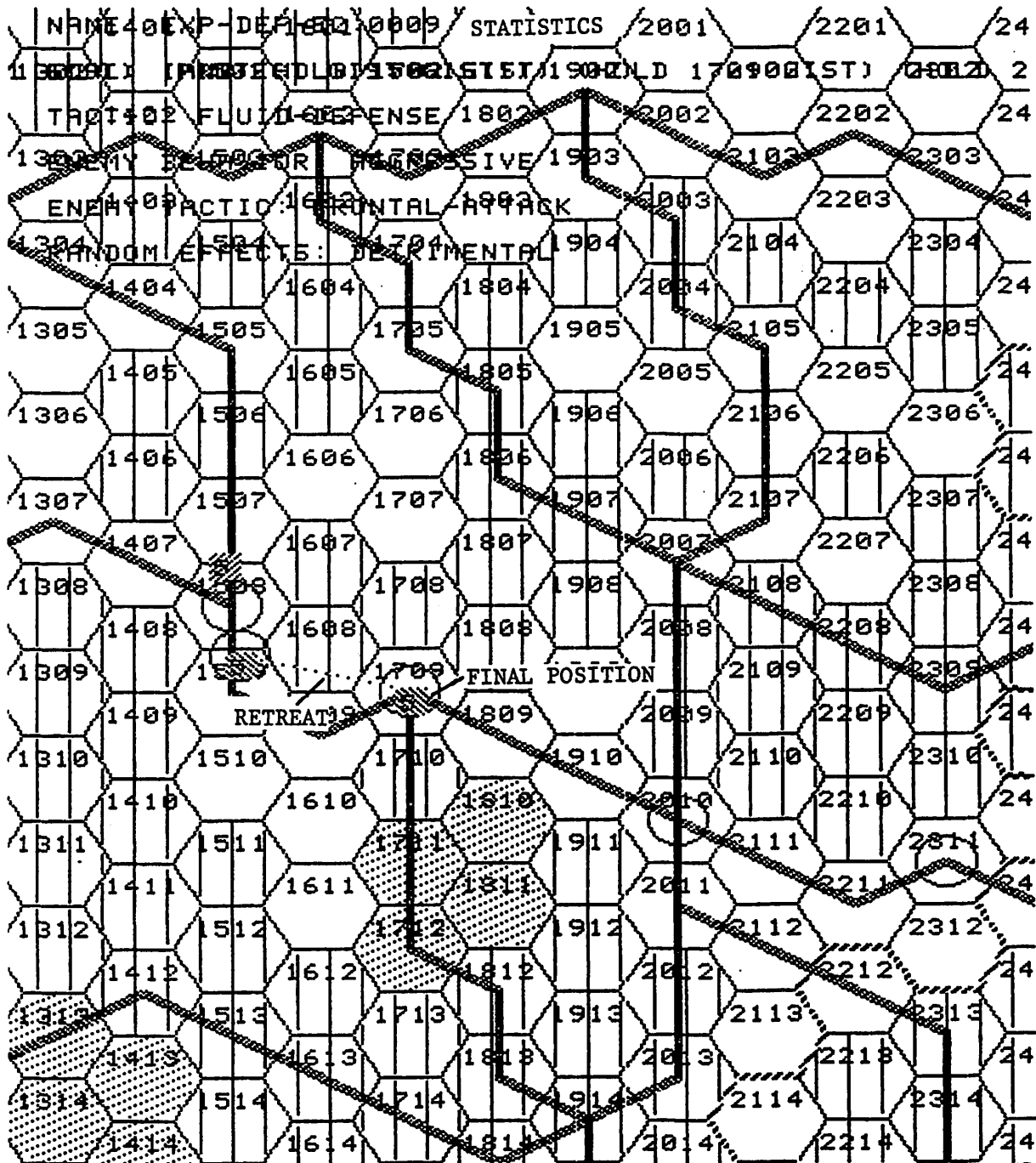


Figure 19. An Alternative Possible World Category from Problem 3 (turn 2). The Union forces have forced the Confederate force to retreat to the second junction.

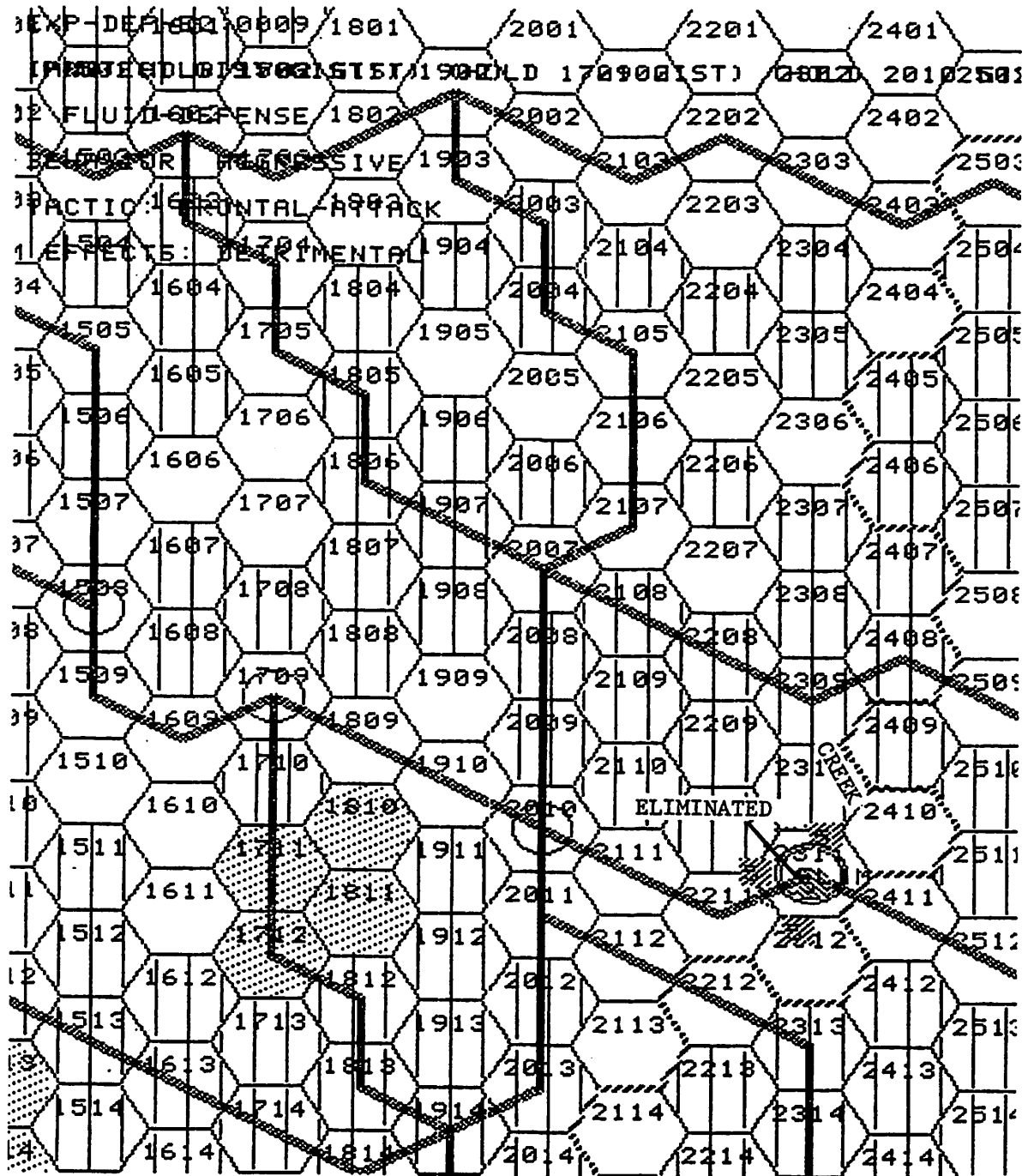


Figure 20. An Alternative Possible World Category from Problem 3 (turn 7). The Confederate force was forced back to the creek where it was surrounded and eliminated.

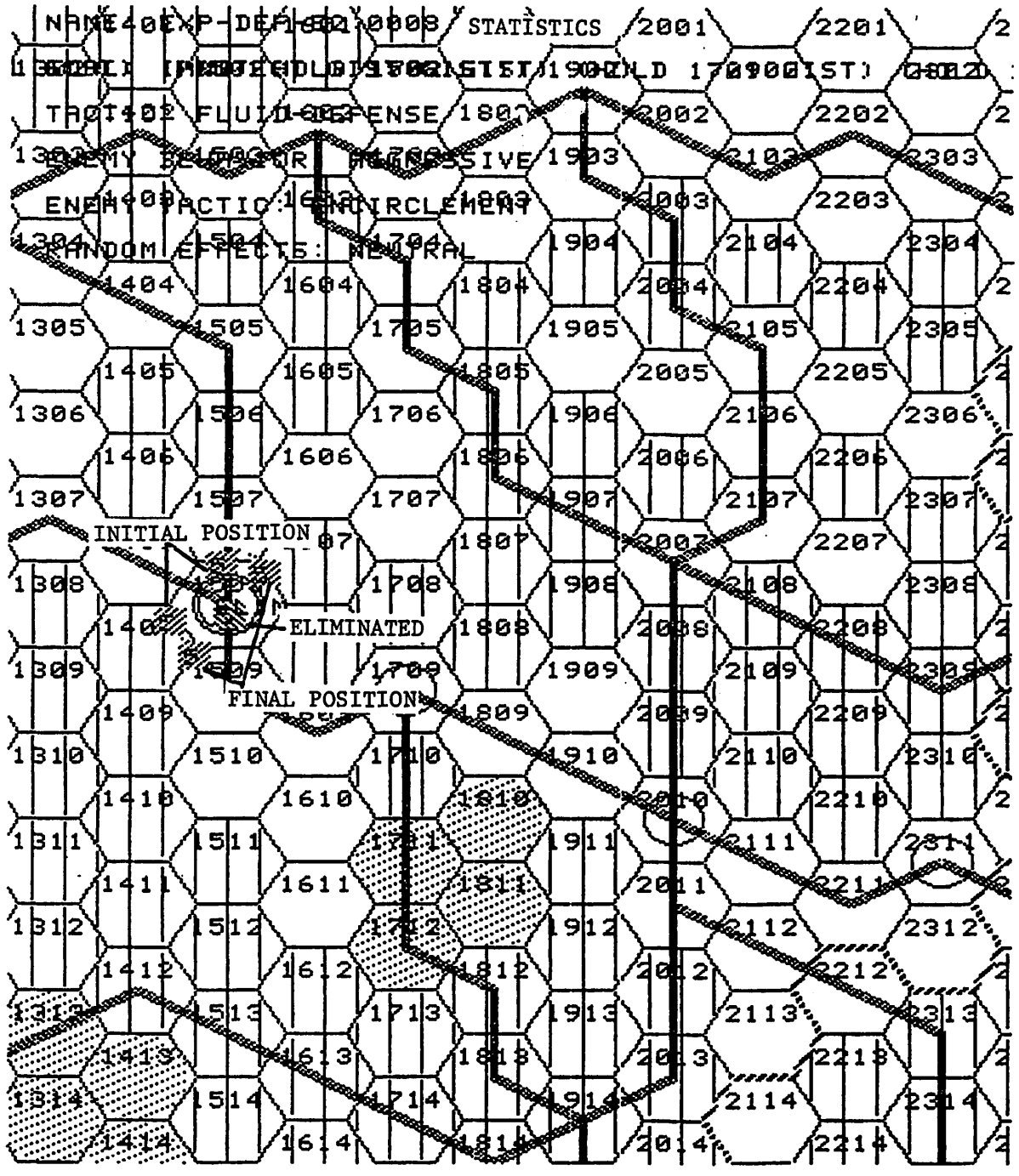


Figure 21. An Alternative Course of Action from Problem 3. The Confederate unit attempts to hold the first junction and is surrounded and eliminated.

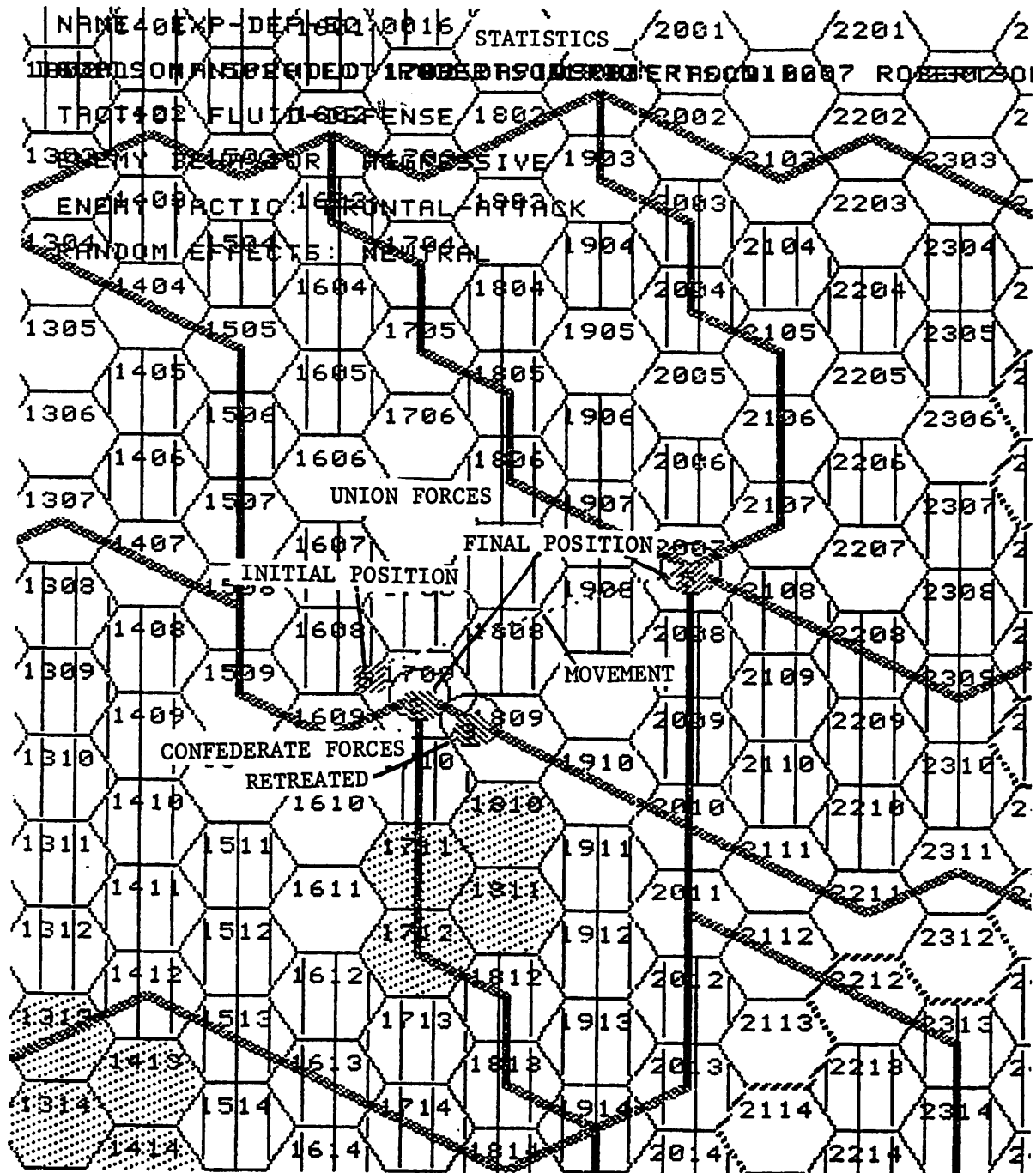


Figure 22. An Example Course of Events from Problem 4. The Confederate forces combined on the first junction. The Union forces attacked and forced the Confederates to retreat. The Union forces then take the unprotected junction.

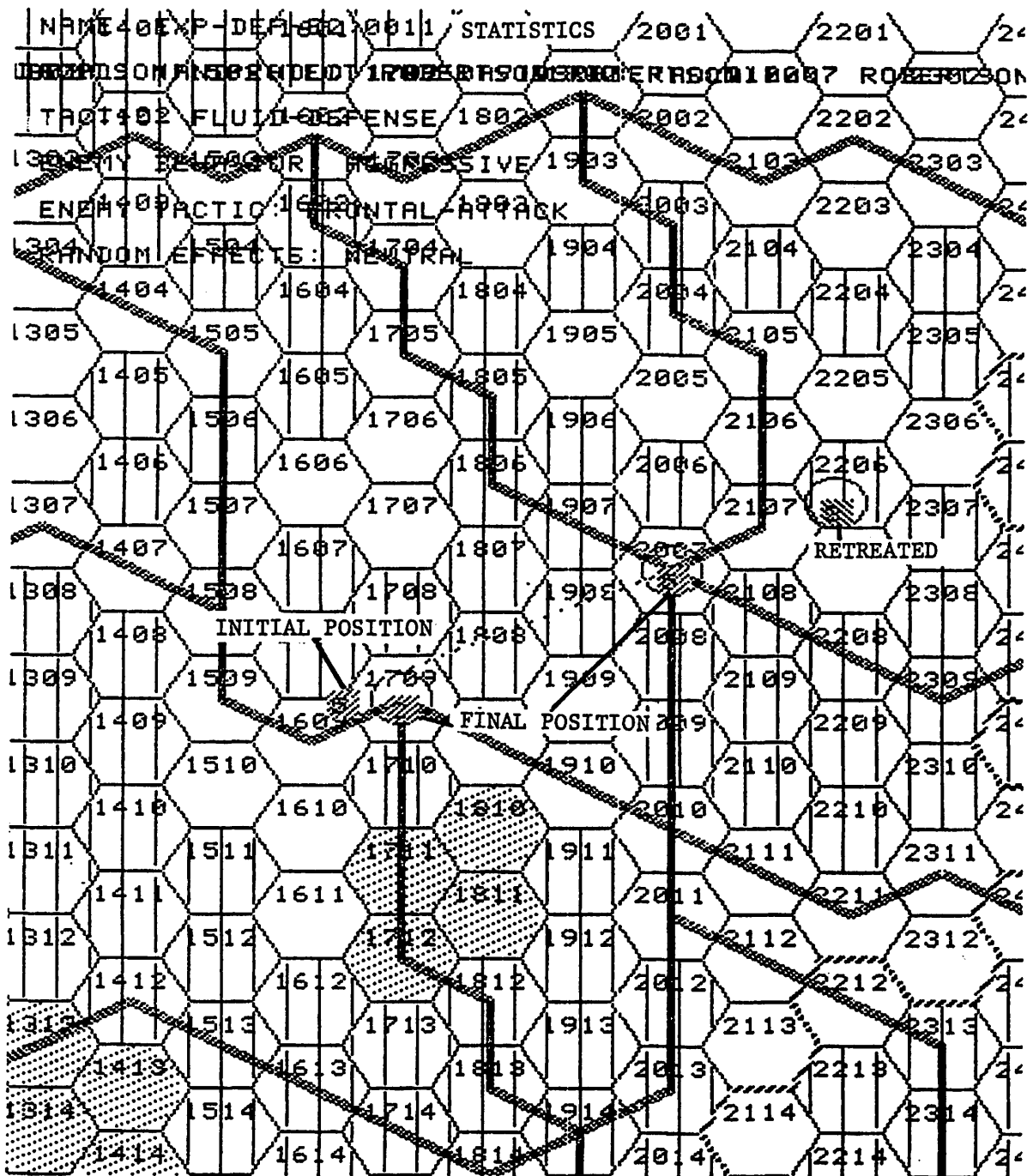


Figure 23. An Alternative Course of Action from Problem 4. The Confederate forces tried to each hold their respective junctions. The Union forces then surrounded and destroyed the first Confederate force (taking that junction) and then pushed the second off the second junction and took it also.

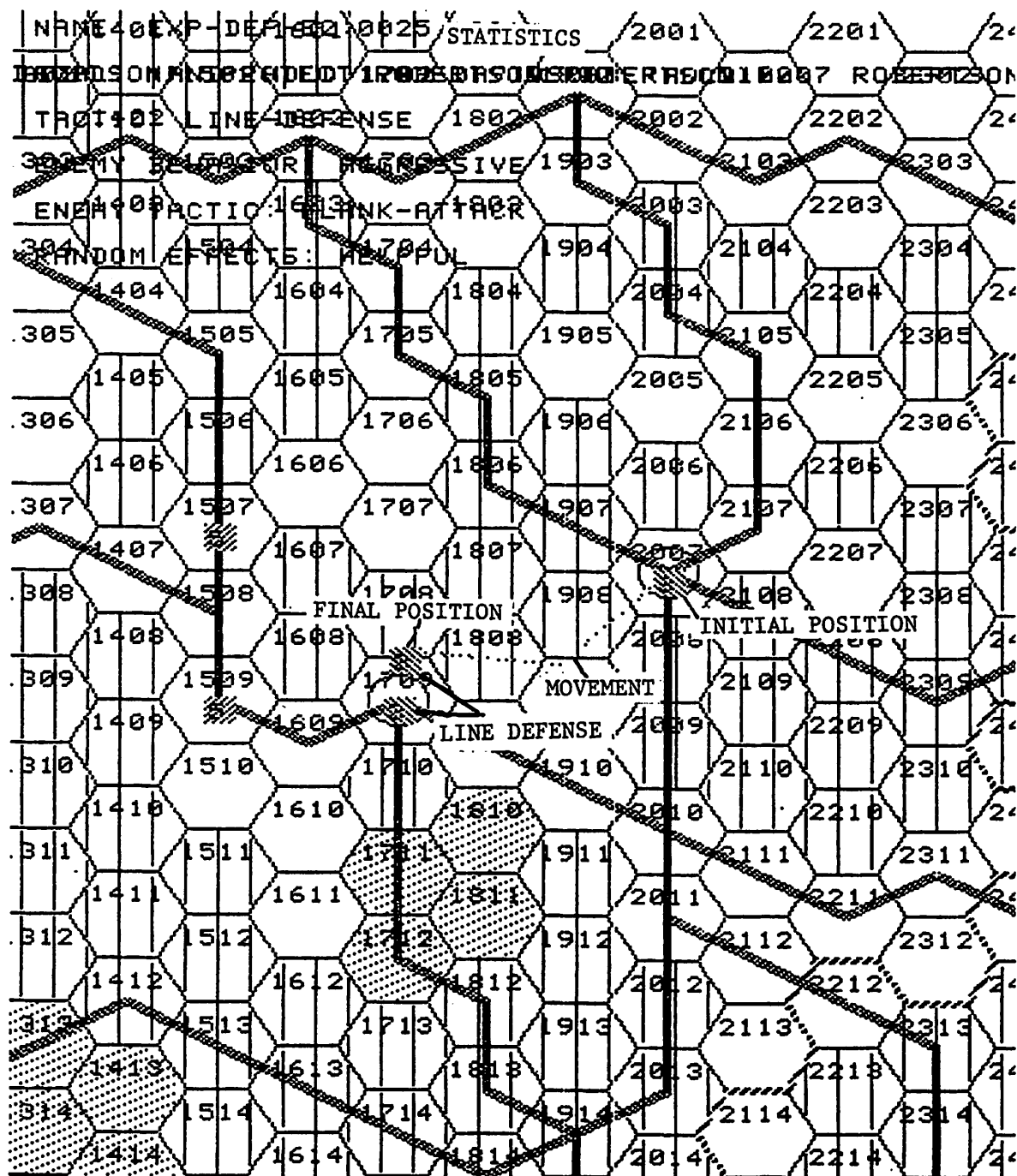


Figure 24. Another Alternative Course of Action from Problem 4 (turn 1). The Confederate forces form a line defense.

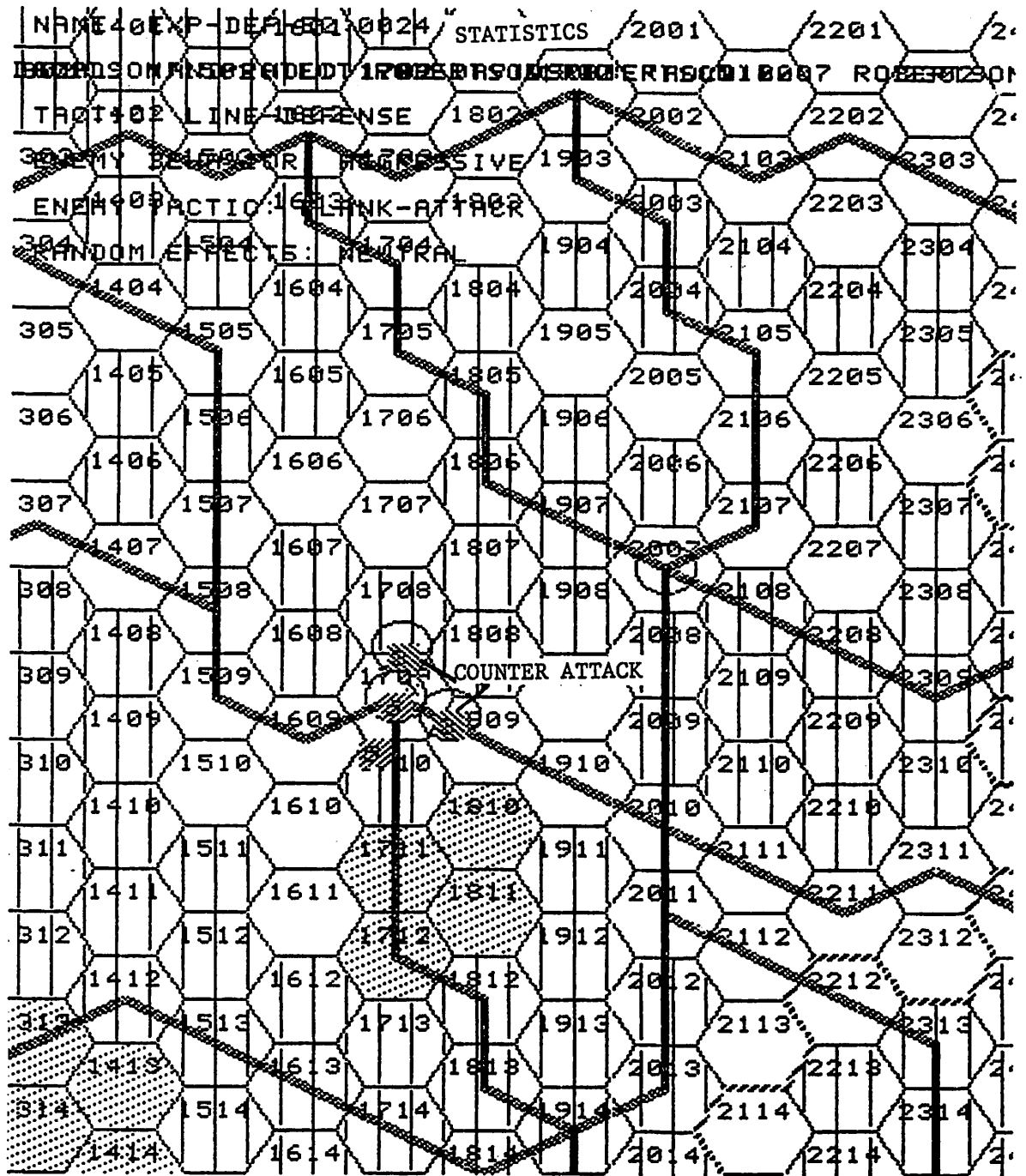


Figure 25. Another Alternative Course of Action from Problem 4 (turn 2). The Union forces attacked and forced the lower Confederate force to retreat, but the Confederate forces attack and retake the junction.

G. The Knowledge Base

This section discusses the knowledge base organization and its components in detail.

At the present time the knowledge base contains 200 experiences. These cover all problems involving a single territorial acquisition goal, one or no defending forces and one or two attacking forces. This includes strength ratios from 3:1 to 1:1, conflicting forces adjacent or separated, enemy forces on or in front of the goal, all friendly tactics, all enemy behavior, all enemy tactics, and all random effects.

Problems involving a conjunction of territorial acquisition or enemy destruction goals with one or two defending forces facing 3:1 friendly strengths are also covered to the complete detail of all friendly tactics, enemy behaviors, etc., including all possible dispositions of forces.

Experiences to aid in simple resource allocation problems are also included. The two resources are territory and units. Often trade offs are necessary between the two when complex goals involving both are present.

G.1. Knowledge Base Organization

The representational structure for the knowledge base is a frame [MINS75]. Figure 26 shows the five basic structures:

GOAL - containing the information about the current goals to be achieved by the system as specified by the user. The slots for this frame were:

name - a unique name for this frame. (Default: current date and time)

type - the type of goal. One of three choices: position, for territorial goals; unit, for unit/group destruction/protection goals; and mixed, for some combination. (Default: position)

value - the numeric value of this goal. (Default: 5)

time - how far in the future in number of turns before this goal is to be achieved. (Default: 0)

specifics - a list of goal descriptors (with optional connectives) specifying what is to be achieved. The format of each descriptor is (connective <desc1> <desc2> ...) where the connectives are either of (and or). Each desc is <verb object agent> where verb is one of (take, hold, destroy, or protect); object is either the location to be taken or held, or the unit/group to be destroyed or protected; and agent is the friendly forces with which the goal is to be achieved. This is the most important slot of this frame. Examples:

(take "1703" gregg) - take possession of hex "1703" with unit gregg.

(and (take "1703" gregg)(destroy union-5 (gregg forrest mcnaire))) - take possession of hex "1703" with unit gregg and remove the unit union-5 from the game using the units gregg, forrest and mcnaire.

Example:

```
goal frame: goal-1
name: "13-NOV-198115:18:21.88"
type: position
value: 5
time: 1
specifics: (take "2007" (conf-g1 conf-g2))
```

SITUATION - the current tactical situation: where the units involved are, their strengths, etc.

name - the unique name for this frame. (Default: current date and time)

date - the game turn in which this situation occurs. (Default: 1)

area-concerned - a semantic tag describing where on the board this situation occurs; e. g., Dyer's Bridge. (Default: nil)

friendly-units - a list of unit/group descriptors one for each friendly unit/group involved. This slot must be in the correct format. Format of each descriptor: <name location strength> where name is the name of a defined friendly unit/group; location is the current location (as a hex number or list of hex numbers, one for each counter) of this unit/group (this may supercede the entry in the cur-loc slot of the unit/group frame); and strength is the strength of this force. The order of descriptors does not matter as the system will rearrange the list to suit itself as the need arises. (Default: the dummy help list ((name1 loc1 strngth1)(name2 loc2 strngth2)) that is supposed to help you remember the format.)

enemy-units - a list of unit/group descriptors one for each enemy unit/group involved. This slot must be in the correct format, which is the same as that for friendly-units above. (Default: the dummy help list ((name1 loc1 strngth1)(name2 loc2 strngth2)).)

map - the map descriptor of the area involved. Not used in the present implementation. (Default: nil)

friendly-losses - a list of the friendly forces that have been lost up to this point in the game. Not used in the present implementation. (Default: nil)

enemy-losses - a list of the enemy forces that have been lost up to this point in the game. Not used in the present implementation. (Default: nil)

friendly-reinf - a list of descriptors similar to the slot friendly-units above showing forces that may be considered as reserves or reinforcements. (Default: nil)

enemy-reinf - a list of descriptors similar to the slot enemy-units above showing forces that may be considered as enemy reserves or reinforcements. (Default: nil)

semantic-desc - a semantic description of the situation.

Example

```
situation frame: sit-2
name: "13-NOV-198115:13:56.26"
date: 1
area-concerned: ("2007" "2010" "2011")
friendly-units: ((conf-g1 ("1911" "1911") 10)
                 (conf-g2 ("2012" "2012") 9))
enemy-units: ((union-1 ("2009") 5)
```



```

                (union-2 ("1809") 5))
map: nil
friendly-losses: nil
enemy-losses: nil
friendly-reinf: nil
enemy-reinf: nil
semantic-desc: (("Enemy" "force"
                    "defending"
                    "trail"
                    "junction"))

```

GROUP - describes the makeup of task groups as defined by the user; in other words, which units are to be considered collected together as a single force.

name - the unique name for this frame. (Default: the current date and time)

units - the names of the units in this group (from the names of the counters that would be in a stack on the board). (Default: nil)

type - the type of units in the group. (Default: inf, for infantry)

cur-loc - a list of the hex number locations of each unit. (Default: nil)

strengths - a list of the strengths of each of the units. (Default: nil)

total-strength - the combined strength of all the units in the group. (Default: 0)

Example:

```

group frame: conf-g1
name: "30-JAN-198214:29:39.21"
units: (gist wilson)
type: inf
affiliation: conf
cur-loc: ("2413" "2413")
strengths: (5 5)
total-strength: 10

```

EXPERIENCE - the heart of the knowledge base: the "memories" of previous encounters with goals and situations, what was done, what happened, etc.

name - a unique name for this frame. (Default: current date and time)

type - epistemological class - su, ref, model, or ce. From Rissland [RISS78]. (Default: su)

tactic - the friendly course of action that this is an example of. (Default: sit)

random-effects - which random effects (helpful, neutral, detrimental) this is an example of. (Default: neutral)

friendly-units - the number of friendly units/groups involved in the remembered experience. (Default: 1)

friendly-names - the names of the units/groups involved. (Default: nil)

friendly-strengths - a list of the situation analysis descriptors for each friendly force involved. Form of descriptor: <type name strength> where type is one of (on, near, next or outside) as defined by the situation analysis function; name is the name of the unit/group frame being described; and strength is the force strength of the unit/group. (Default: nil)

disposition - the tactical arrangement of the friendly forces. (Default: col, for column)

location - the locations (hex numbers) of the friendly forces. (Default: nil)

terrain - the type of terrain in which the experience takes place. (Default: clear)

enemy-units - the number of enemy units/groups involved. (Default: nil)

enemy-names - the names of the enemy unit/group frames involved. (Default: nil)

enemy-strengths - a list of force descriptors from the situation analysis for each enemy force involved. Form: <type name strength> where type is one of (on, between, near or outside); name is the name of the unit/group frame; and strength is the strength of the force. (Default: nil)

enemy-tactic - the tactic the enemy is using. (Default: sit)

enemy-behavior - the mode of behavior the enemy forces are assumed to be in. (Default: neutral)

enemy-disposition - the tactical arrangement of enemy forces. (Default: line)

enemy-location - the hex number locations of the enemy forces. (Default: nil)

distance - the number of turns of unrestricted movement it would take for the closest friendly unit/group to reach one of the goal locations. (Default: nil)

destination - the hex number location of the goals at the beginning of the course of action in the experience. (Default: nil)

engagements - a list of pairs of force numbers that engaged in combat during this experience. e. g., if friendly force 1 fought enemy force 2 then the pair (1 2) would be in the list. (Default: nil)

goal - the list of goal descriptors that were to be achieved in

this experience (similar to the goal frame's specifics slot).
(Default: nil)

coe - the course of events of the remembered experience as a list of ply descriptors. Each ply descriptor is a list of movement and combat descriptors depicting what happened. (Default: sit)

Example:

```

experience frame: exp-e1-front-goal:0026
name: " 6-JAN-198200:22:38.12"
type: su
tactic: frontal-attack
random-effects: helpful
friendly-units: 4
friendly-names: ((gregg davidson) (mcnair robertson))
friendly-strengths: ((next (gregg davidson) 7)
                    (next (mcnair robertson) 7))
                    ;the forces were adjacent to the enemy

disposition: line
location: (("1909" "1909") ("1910" "1910"))
terrain: (clear)
enemy-units: 1
enemy-names: ((union-8))
enemy-strengths: ((between 1 2) (union-8) 5)
                  ;the enemy was between the friendly
                  ;forces and the goal

enemy-tactic: frontal-attack
enemy-behavior: aggressive
enemy-disposition: line
enemy-location: (("1809"))
distance: 1
destination: ("1508")
engagements: (((1 2) 1))
goal: (take ("1508") ((force 1) (force 2)))
      ;an acquisition of territory goal
coe: (((combat (1 2) 1 (enemy-retreat aa:))) ;ply 1
      ;both friendly forces attack the
      ;enemy force and cause it to retreat
      ((combat 1 1 (friendly-retreat aac))) ;ply 2
      ;the enemy counterattacked and
      ;forced the retreat of friendly
      ;force number 1
      ((move 1 (to (right-flank -1))) ;turn 2 (ply 3)
      ;begin to surround the enemy unit
      (move 2 (to (left-flank -1)))
      ;complete surrounding of enemy
      (combat (1 2) ;attack again,
      1 ;this time, enemy is

```

```

      (enemy-eliminated aac));eliminated
nil      ;turn 2 (enemy has no move - ply 4)
((move 1 (to (on goal)))));turn 3, ply 5
      ;take goal hex

```

SCENARIO - actually a collection of information. It is a list of five elements:

1. the goal frame describing the goals to be achieved; as supplied by the user
2. the situation frame describing the conditions under which to achieve the goals,
3. the list of relevant instantiated experiences. This is organized as a list of lists. Each top level element is a segment as defined in Chapter Three: a set of projections for a given course of action in a given possible world category. The number of experiences included depended on the difficulty of the problem. The simple problems had as few as 15 experiences and the complicated problems as many as 75.
4. the list of solved queries and
5. the list of unsolved queries.

Figure 26: Detailed Structure of the Knowledge Base

G.2. Knowledge Base Details

Specifically the experiences in the knowledge base have been designed to provide tactical advice on any of the following problems:

- o Acquisition of simple territorial goals with no enemy opposition.

- o Acquisition of a territorial goal with one friendly force against one defending enemy force with 3:1 odds.
- o Acquisition of a territorial goal with one friendly force against one defending enemy force with 2:1 odds.
- o Acquisition of a territorial goal with one friendly force against one defending enemy force with 1:1 odds.
- o Acquisition of a territorial goal with two friendly forces against one defending enemy force with 3:1 odds.
- o Acquisition of a territorial goal with two friendly forces against one defending enemy force with 2:1 odds.
- o Acquisition of a territorial goal with two friendly forces against one defending enemy force with 1:1 odds.
- o Acquisition of two territorial goals with two friendly forces against two defending enemy forces with 3:1 odds on each enemy force.
- o Destruction of one enemy force with one friendly force with 3:1 odds.

- o Destruction of one enemy force with two friendly forces with 3:1 odds.
- o Destruction of two enemy forces with two friendly forces with 3:1 odds against each enemy.
- o Holding a section of road with a single friendly unit against two enemy units who have 2:1 odds while trying to protect that unit from being destroyed.
- o Holding a region of the map with two friendly units against two enemy units with 3:1 odds against each of the friendly units that are also to be protected.

The length of the courses of events in the experiences was on the average four to six ply long. Some were as short as two ply and some as long as eighteen.

H. System details

Figure 27 shows a functional diagram of the scenario generation by retrieval and modification of experience mechanism.

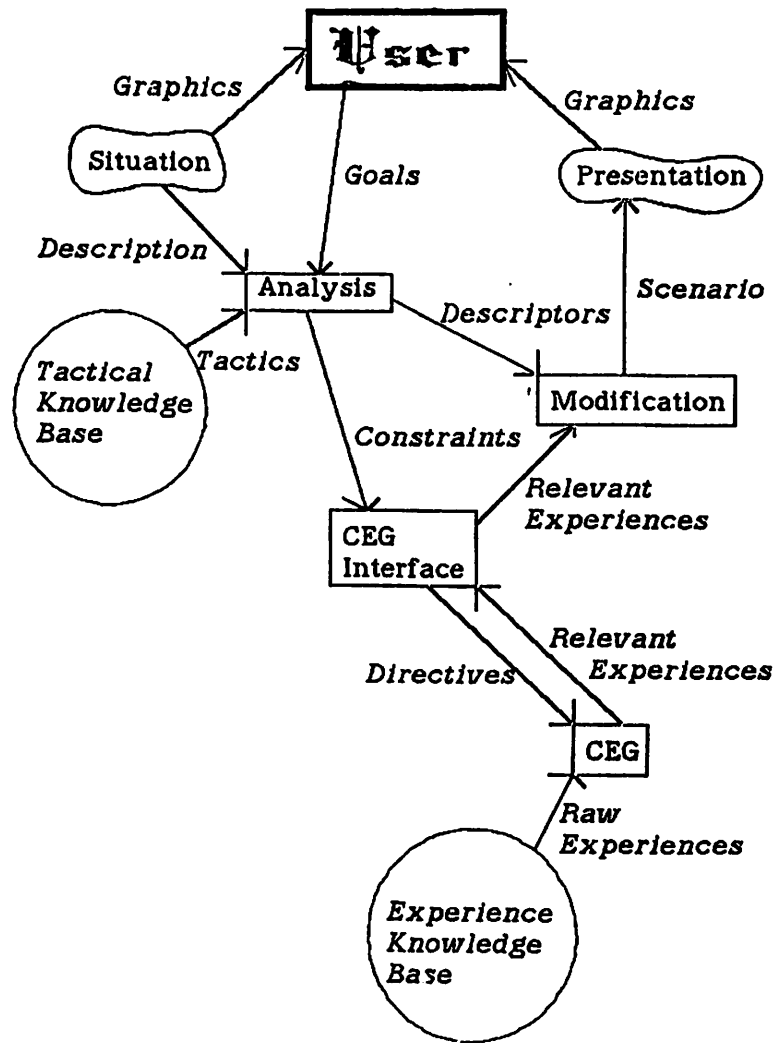


Figure 27. The Scenario Generation Mechanism.

H.1. Analysis

The Analysis Subsystem uses the current situation, the goal-set of the user, and knowledge about the tactics available in the domain to create constraints that will select relevant experiences, and descriptors that will modify those experiences to fit the current situation exactly. It examines the arrangement of friendly and enemy forces in relation to the goals of the friendly process.

It then generates descriptors for each force. These descriptors are placed in constraints and combined with the goal description constraints and possible world category constraints to form queries. These queries are used to locate experiences of similar situations. Once the relevant experiences have been recovered these descriptors are used to modify model experiences, or instantiate start-up and reference experiences, to fit the current situation exactly.

H.2. Constrained Example Generation Interface

The Constrained Example Generation Interface Subsystem directs the Constrained Example Generation (CEG) system [RISS82] to search the knowledge base. This is done by presenting the queries formed by the analysis routine. All experiences that match any of the queries are collected. It also includes the judgement routines needed by the CEG system to evaluate the closeness of each potentially relevant

experience to the current situation.

H.3. Instantiation

The Instantiation Subsystem controls modification of relevant experiences to fit the current situation. This is done for most experiences by taking the descriptors from the Analysis routine and filling in the corresponding slots. For model experiences, however, a set of context parameters including the possible world categories and the course of action must be set up. Then the course of events slot of the experience, which is a LISP PROG expression, is evaluated. The result is the actual course of events placed in the scenario.

H.4. User Presentation

The User Presentation Subsystem consists of a series of graphic display routines that draw for the user each course of action from the instantiated experiences. It includes an interface that allows the user to move about the possible futures as desired, e. g., the user can ask to see all detrimental worlds, or those in which the enemy was aggressive. The system also includes facilities to show situations, goals and individual experiences from the knowledge base.

The appendix contains a detailed description of the inner workings of the system.

I. Support Software

In addition to the system software described above the following facilities were developed:

1. A symbol management executive. Keeps track of goals, situations, scenarios, tasks, and forces. Performs creation, deletion, copying, printing, and modification of the knowledge base elements, including file maintenance.
2. A frame maintenance system. Performs creation, printing, and modification of frame data structures including application of the lisp function editor to arbitrary frame facets; as well as retrieval and replacement of frame facets.
3. A language for describing courses of action and a system to instantiate as specific unit locations and translate to graphical output. Includes movement to arbitrary semantic positions such as "front", "left-flank" etc, combat and combat results.
4. A syntax for describing situations and a method of semantically capturing relationships between forces and between forces and goals automatically.

C H A P T E R VI
RESULTS, CONCLUSIONS AND IMPLICATIONS

This chapter discusses the results of this work including its contributions and accomplishments. First, results from the work done in the errand running domain are presented, second, results from the conflict simulation game domain, and finally a discussion of the conclusions and implications to be drawn from this work.

A. Results from the Errand Running Domain

Work in the errand running domain was an important first step in developing and testing the theory of scenarios. In particular, it tested:

1. The usefulness of scenarios.
2. Use of scenarios as a presentation scheme for a Tactical Assistant.
3. The idea of categorical analysis. It tested the practicality of treating random processes symbolically in scenarios.

4. Use of simulation methods to build scenario segments.
5. Use of retrieval and modification of experience to build scenario segments.

It also allowed a comparison of the two methods to be made.

A.1. The Usefulness of Scenarios

The main goal was first to try out some of the ideas in a restricted domain; second, to compare two methods of scenario segment creation; and third, to actually build something useful.

It is difficult to show the usefulness for the simulation based system since it required months of work for a one shot system. However, it was only meant to be a demonstration system. I feel that within the limitations of the host machine (a CYBER 175) and the resulting limit on detail, the scenarios produced are a good look at the future.

The usefulness of generating scenarios using the Constrained Scenario Generation system is another story. This system was designed to be as domain independent as possible. Thus the application of this system to errand running was very easy, taking only a week to complete.

The conclusion is that the range of possibilities suggested by the set of scenario segments produced allows the user to get a good idea of what could be in store and allows planning decisions and goal specifications to be reviewed and restructured without having wasted any time actually performing tasks.

A.2. Use of Scenarios in a Tactical Assistant

The purpose of a Tactical Assistant is to take the high level goals of a planning or user process, develop the necessary low-level or tactical information needed to perform the specified tasks, and return to the user an outline of the possible courses of events. The scenario is an excellent vehicle for this task. The most important reason is that the scenario contains a collection of possible world examples. This allows the user a view of the alternative futures.

The scenario by the Tactical Assistant is a form of abstract plan that can then be checked for its consequences. Since the scenario contains a collection of possible world examples, the user can examine the results of various assumptions about the future behavior of the various foreign processes.

A.3. Use of Categorical Analysis

Categorical analysis was used to select and control the possible worlds considered in both the Tactical Assistants built. This was described in Chapter Four for errand running and Chapter Five for conflict simulation games.

This part of the work demonstrated the success of categorical analysis. It allowed the blur of future worlds to be divided up cleanly into manageable portions, while still giving a complete picture. The other part of the use of categorical analysis in the errand running domain was to see if categorizing random processes symbolically as opposed to using utility theory or some form of probabilistic presentation was helpful.

I decided that for most people planning errands it doesn't really matter if, for example, by going to a particular store at a particular hour one has an 'x' chance of being able to buy a newspaper giving a utility of 'y' for that trip, rather what is important is the result: If they are sold out of newspapers one will have to go somewhere else to buy one.

In using the simulation method, the random processes are handled internally on a probabilistic basis - there are initial estimates and conditional modifiers. The important point is that if the estimated chances of the event happening exceed the level at which the expert

has decided to worry about such things then the event is shown to the user as having happened and its effects noted. If the user decides such a level of concern is prudent then planning decisions can be made accordingly. Otherwise the posted effects can be ignored.

The user is not shown the underlying probabilities. No indication is made in the presentation of how likely or unlikely the course of events shown is. Instead the focus is on the goals of the user and the effects the actions of uncontrolled processes can have on them. If something is especially crucial, and there is a chance a random event could mess it up, the user should be told.

In using the retrieval and modification of experience method, experiences of similar courses of events in similar world categories are selected directly to outline the possibilities of the current situation.

The relative levels of what is likely or unlikely is fixed by the implementor under the advice of the domain expert. For example, the bank was assumed to always have money. There is a chance, very small, of there being a run on the bank; however, the user is not informed. These and other events such as a meteor hitting the user's car are considered too unlikely for consideration. This decision is in a sense an arbitrary one by the domain expert, since a meteor striking the user's car would certainly prevent accomplishing some of the

goals. But at some level of probability a cut off must be made below which events will no longer be included. As the system is used the various probabilities can be tuned or additional events or processes added.

A.4. Building Scenario Segments by Simulation

The form of simulation used in this Tactical Assistant is not exactly the same as normal simulation methods. Instead important actions were chosen on the basis of the goals of the user and courses of events built around them.

There was no tree search or search of any kind as is normally associated with planning in complex domains. The most important point of a tree-less simulation was that it allowed the system to avoid the processing involved in exploring multiple choice pathways. This was done by focusing on the goals of the user and having more knowledge in the planning heuristics and their application.

The simulation technique used in this project was most similar to an event-driven simulation as opposed to a discrete simulation. The difference is that in our method the temporal order of event posting and course of action organization was arbitrary and controlled by the goals given to the Tactical Assistant, rather than the processes operating in the world.

A.5. By Retrieval and Modification of Experience

The generation of errand running segments by retrieval and modification of experience was meant to demonstrate the domain independence of the Constrained Scenario Generation System rather than to say anything new about scenarios in the errand running domain. To this end the results were very dramatic. It turned out to be surprisingly easy to take out the parts of the system specific to conflict simulation games and replace them with parts specific to errand running. In fact, as mentioned above, it only took a week to perform the entire transformation.

B. Results from the Conflict Simulation Game Domain

The main component of the research work is the Tactical Assistant. A Tactical Assistant for the game Chickamauga has been implemented that meets all the goals proposed. The proposed work of this thesis included research into:

1. The construction of a Tactical Assistant for the conflict simulation game Chickamauga.
2. Use of Retrieval and Modification of Experience to build course of event projections.
3. Full analysis of possible courses of action and foreign process effects with scenarios using categorical analysis.
4. Construction and Maintenance of a Knowledge Base of tactical experience.
5. Some demonstration of the system's utility.

B.1. Conflict Simulation Game Tactical Assistant

The domain of conflict simulation games is extremely complex. The creation of a detailed plan for a complete game is beyond the current Artificial Intelligence technology. However, a tactical assistant is not designed to return a complete solution. Instead, by

previewing the future an informed choice of course of action can be made. The conclusion is that the use of a Tactical Assistant is a useful approach for these sorts of complex domains.

B.2. Use of Retrieval and Modification of Experience

In implementing a system to generate scenarios by retrieval and modification of experience we demonstrated not only that it was possible to generate scenarios by this method but also that this approach was generally useful.

This is shown by the domain independence of the Constrained Scenario Generation System. The conclusion is that the use of the Constrained Scenario Generation System is useful in complex domains for which a large body of experience is available, and in which it may be difficult to generalize and abstract rules.

B.3. Use of Categorical Analysis

It was only through the use of categorical analysis that anything was able to be said about the problems in the conflict simulation game domain. The results showed the need for such a tool in such complex domains. For example, in problem 2, the use of categorical analysis allowed the user to examine examples from 50 possible worlds rather than attempting to consider the thousands of tree search branches

available.

B.4. Experiential Knowledge Base

The results of the implementation show the practicality of representing tactical knowledge as experiences. It was relatively easy to add information about a new problem once the domain expert was familiar with the format. The only problem was in the slowness of the computer in updating large data files.

It was useful to categorize the experiences into three classes (after Rissland [RISS78]): start-up, reference, and model. The start-up experiences were simple cases with few units or complications. Reference experiences were the "classic" examples of some behavior or tactic, as well as what Rissland called "counter" or "anomalous" examples: novel or unusual experiences. Model experiences were those situations that were familiar enough to the expert that most of the particulars could be abstracted away, to be filled in later by the context in which it was selected.

B.5. Demonstration

Although not formally statistical the results of the human study were persuasive. The system was of benefit to novice and expert alike. The problems presented to the subjects were all fairly simple,

but the resource problems (problems three and four) were more difficult than the others. It was on these problems that the system appeared to help the gamers and experts. Thus it would seem that this kind of Tactical Assistant system would be of even more help in still more difficult problems involving more complex goal structures or a complete game.

The comments that were made were nearly all positive, the few negative comments were requests for more features that were not considered in the original design proposal.

C. Conclusions and Implications

This section discusses the conclusions and implications of this work. These are grouped into the following areas: The use of scenarios, knowledge representation, the meta-structures used to produce scenarios, the handling of foreign processes, the retrieval and modification of experience paradigm, and the use of categorical analysis.

C.1. Use of Scenarios

This work is a step towards finding a solution to decision making in complex environments. Together with a system to actually make decisions based on review of scenarios a potentially very useful planning tool can be created. The old search paradigm consists of trying to simultaneously generate and evaluate future worlds while choosing a course of action to achieve the desired goal world. With the scenario approach the user is shown a series of examples, each representing an entire class of future worlds. The task is then only to choose the course of action to pursue.

The scenarios presented to the users by a Tactical Assistant give a range of things to think about. There may be courses of action or foreign process effects that they might not have thought of. It is up to the planner to decide how much worry to give such items. Thus the

Tactical Assistant is of more use to the user who has some familiarity with the domain than one who knows nothing.

The usefulness of scenarios for a given domain is dependent on the complexity of that domain and the difficulty in generating a knowledge base of experience for it. Given that experts in a domain are available and can express their memories in a suitable manner, the retrieval and modification of experience method of generating scenarios could be applied to any domain where the profusion of possible future worlds limits analysis of future outcomes. In addition, a Programmer's Guide was written to aid interfacing an arbitrary domain to the Constrained Scenario Generation System.

C.2. Comparison of the Two Methods

This section compares and contrasts the two implemented methods of generating scenario segments. Most of the differences are based on the two different methods of representing the knowledge in the domain. Although both methods have uses in particular contexts and there are benefits and drawbacks for each the main conclusion is that for some of the more complex domains the representation of knowledge by experience may be better.

The main point is not to suggest that one method should be used to the exclusion of the other. Rather it is clear that there are representational needs for both. For example, in the Constrained Scenario Generation system the tactical knowledge is in the form of rules while the experiences are in the form of frames. One can think of rules as generalizations and abstractions of experiences, able to be applied to arbitrary situations.

C.2.1. Implementation Differences

One problem with bringing up a rule based system is tuning the IF portion of the rules so that the correct rules will fire. Retrieval and modification of experience has an analogous problem, that of tuning the tactic/situation analysis routines to generate the correct constraints to capture the correct experiences.

There may be an inherent bias in the set of experiences provided by the expert. The choices of likely/unlikely and the behaviors and tactics for a given situation and the completeness of the experience base can only really be tested by running the system on real problems and seeing how it does.

C.2.2. Incorporating a Novel Situation

Once the rules or the analysis routines have been tuned, what should the system do when faced with a novel situation that cannot be handled by the existing knowledge? With rules the situation must be analyzed to figure out what about this situation makes it different from the situations that the rules cover - what can be generalized and abstracted from this experience. However, using the retrieval and modification of experience knowledge representation scheme, this novel situation and experience is exactly what the knowledge base is to contain - it need merely be inserted into the knowledge base as it is.

What happens if the system encounters a situation that is so unusual that it can not be described by the rules and predicates or the situation analysis routine? What does this mean for the information in the knowledge base? Is it still valid? Using retrieval and modification of experience there are two possibilities: either new descriptors must be added to the situation description language or the language must be completely redesigned. In the first case there is no real problem. The new descriptors are used for the new type of situation, the old descriptors for the old type.

An instance of the second case actually occurred during this research. A knowledge base of experiences was built to solve simple problems similar to the ones given to the human subjects as problems

one and two. The decision was then made to see if the system could be extended to cover resource problems such as those in problems three and four. As these new problems were being examined it became apparent that the previous way of looking at situations was not completely correct, and that the situation description language should be rewritten. A new situation analysis routine was also written to generate the new descriptors. The new experiences could now be added.

But what of the old experiences? Were they to be thrown away? No, the information in the courses of events depicted was still valid, it just needed to be reinterpreted. This was done very simply: The new situation analysis routine was run on each old experience, and since each contains information about the arrangement of forces and the goal-set that was to be accomplished, each could be redescribed using the new descriptor set. These new descriptors then replaced the old descriptors in the experience frame, and all the old knowledge was still accessible.

C.3. Knowledge Representation Results

In using the retrieval and modification of experience form of knowledge representation, there were a number of different ways and places that different pieces of domain specific knowledge could be implemented. These were: the domain knowledge used in scenario display, tactical and situational analysis, and the actual experiences

in the knowledge base. Great care was taken during the design and implementation of this system to keep all domain knowledge restricted to these areas and out of the scenario generation mechanism itself.

This allowed examination of how choices of knowledge implementation at one level constrained choices at the other levels. These choices included not only what specific knowledge to place at what level, but also choices of abstraction of domain knowledge at different levels and what types of knowledge worked best at which level.

An example is knowledge about tactics. For instance, the top level display needs to know about encirclement in order to draw it correctly, but this can be done by letting it know about such things as flanks and fronts. Then the knowledge about which tactic means move to the flank and which to the front can be placed in the course of events slot of the experiences. But then how to tell when to refer to which experience? This knowledge is placed in the tactical analysis routine.

In addition the following problems involved in generating scenarios by retrieval and modification of experience were solved:

1. The form of scenario to be produced: what would be useful and understandable by a user of the system.
2. The choice and form of constraints used to query the knowledge base so that all pertinent experiences are retrieved with a minimum of extraneous information, yet allow
3. Instantiation or modification of the experiences to fit the exact goal and situation in the original problem posed.
4. The form, number and details of the experiences in the knowledge base so that all courses of action and foreign process effects would be demonstrated yet allow practical maintenance.

In other words, the experience base had to be general enough to answer questions yet specific enough to allow manageability. The modification routines had to be complex enough to transform an arbitrary experience into a scenario segment yet not create spurious or inapplicable segments; the constraints broad enough to ask every question yet precise enough to get answers. And finally, the scenario produced complex enough to provide the user with a complete view of the future yet easy to understand.

The solution to the second point, the form and choice of constraints involves deciding what features in the tactical problem (goal/situation) are important, how to describe those features, how to notice and select similar features in the experiences, how to judge exactly what is similar, and how to define the proscriptive course of action and foreign process constraints.

The problems of points two, three and four, the form and choice of constraints, the methods of modification of experience and exactly what information about each is in the knowledge base of experience, actually had to be solved simultaneously. These facets have gone through numerous iterations of change in style and content before reaching their present configuration. Once the form of the user presentation was chosen, the first dozen iterations were on paper; followed by implementation and testing, which of course pointed up various unforeseen problems, necessitating changes, more experiments, etc.

Each tactical experience in the knowledge base represents a "memory" of some problem encountered in the past. One of the research issues was exactly what features of that previous encounter should be "remembered". This problem was resolved simultaneously with the three issues described above. The format arrived at allows start-up experiences to have simple values in the various facets as well as model experiences to have complex conditional values and reference

experiences in between. The knowledge base contains some 200 experiences. A scenario will contain between 15 and 75 experiences depending on the complexity of the problem. There is an onto relationship between the knowledge base experiences and the modified experiences shown the user: It is possible for one knowledge base experience to be valid in more than one context.

An additional problem involved deciding at what level of abstraction to represent the experiences. For the current version the experiences describe courses of events at the level of the individual piece in the game (the units). Thus the events in the course of events were such things as the movement of a unit, combat between units and the results of combat on the units. The average experience was five ply long, but the median is nine ply due to the length of some of the experiences relating to the resource problems.

Further research is necessary before specific recommendations about the distribution of knowledge for an arbitrary domain can be made.

C.4. The Constrained Scenario Generation System

The Constrained Scenario Generation system should be applicable to a wide variety of domains, given that the domain specific knowledge of scenario display, tactical and situational analysis, and the actual

experiences could be collected and implemented.

By showing the usefulness of scenarios, it is felt that a case has been made for scenarios as an aid in complex domains. Even though a domain such as oil well analysis or program synthesis is far removed from planning, and is not concerned with "the future" shown by planning scenarios, solving such problems could be thought of as having a template or schema with slots, values or functional elements to fill, a scenario could be used to show a series of "snapshots" of that template on its way to an attempt to instantiate the solution to the problem.

C.5. Handling of Foreign Processes

Another important result of this work is the capture of the essentials of the foreign processes by making assumptions about their behavior.

This allows statements to be made about their actions without really having a clear idea of their exact goals. This result has implications for numerous planning and problem solving environments in which the exact intentions of foreign processes are unknown, for example, command and control problems or threat assessment.

C.6. Categorical Analysis of the Future

The final conclusion of this work is that meaningful examinations of future possibilities can be made by categorical analysis. The relative ease with which futures can be examined using this idea in this complex domain suggests its applicability to other domains.

The ability of categorical analysis to "collapse" numerous possible worlds in a few prototypical worlds as part of the assistance provided by a Tactical Assistant has been shown to help the users of the Tactical Assistant.

D. A Potential Application

Potential applications of scenarios include the domains of oil well analysis, threat assesment, command and control and strategic planning. Another potential application of this work is in Emergency Room medicine. The goal of the system would be to take the patient's symptoms and statistics and suggest possible causes and potential results of choices of treatments.

In addition to simply providing more information for the physician, the system would be able to show the potential course of events given assumptions about the way the unknown processes might act on the patient given a choice of course of action by the physician. For example, it could show the consequences of treating for disease a

when the problem is really disease b, which sometimes looks similar.

Additionally, the system could point out to the physician that there is a chance the problem is c, something they might not have normally thought of. The system would be able to show possible futures in which it turned out symptom x was masking the real problem that could be detected and corrected by a given treatment t.

All of this information would, of course, be provided by experts in the field. Much of the skill of an Emergency Room Physician is in the collection of experiences they have accumulated during their practice. A potential problem for this application is defining the way in which situations will be described for the machine to use. This is because a large portion of the data available to the physician is sensory: how the patient sounds, looks, and even how they smell. Additionally, it may be difficult to express most of this information symbolically, since again it is sensory experience that may never have had to have been expressed verbally before. However, it is clear that since the retrieval of experience plays an important part in diagnosis and treatment, and there is such a need for experienced physicians, that some sort of approach similar to that presented here could be pursued.

E. Future Work

There is much research still to be done in the area of planning and problem solving in domains with foreign processes. This section outlines some of the work that is yet to be done that could build on this thesis. First, those projects that could be undertaken by a Master's candidate or by someone with a year or so of time to work on them. Then the deep research projects that still remain are discussed.

E.1. Work to do Given a Year

Some of the additional work that could be done in the system given a year or so more work could include:

1. Commentary during the scenario display to aid user understanding.
2. Explicit expression of similarities and differences among the various future views being presented to highlight the consequences of tactical decisions made. At present this information is available but the users must make their own inferences. This could be done as a simple comparison of force strengths and positions before and after each course of events.

3. A larger knowledge base capable of handling more complex problems. This may potentially cause another rewrite of the situation description language, which is a slightly larger problem.
4. There is a need for a stronger relational framework for the experiences to highlight the relationships among them, for example, the situation produced at the end of experience-17 might be the beginning of experience-107. Larger problems might be solved by "splicing" together various simpler experiences. And potentially playing entire games.
5. Consideration of terrain effects. This would necessarily require some changes in the situation description language.
6. Explicit examination of the strategic effects of tactical results. Again, at present the user must infer these effects for themselves. This would be done by examination of the situation before and after the course of events with a method of supplying information about the strategic goals of the user.
7. Additional information about how "likely" each course of events is could be shown. In some cases this is difficult to do since the course of events might represent a series of fortuitous blunders by either side that could happen often or rarely depending on the players.

E.2. Major Research Projects

Major research is still to be done in the following areas:

1. Examination of competing in addition to unfriendly and random foreign processes.
2. Other friendly processes uncontrolled by the process generating the analysis.
3. An analysis of similarities and differences among representations of knowledge to generate scenarios in different domains.
4. A complete theory of how an arbitrary domain should be broken down for implementation - what parts as rules, experiences; how to represent the relationships among the knowledge; what is declarative, what procedural; in other words, the fundamentals of knowledge engineering.
5. Development of a decision engine that would examine the futures presented by the Tactical Assistant and choose courses of action based on that analysis (a "commander"). In its full form such a device would allow research into such esoteric concepts as surprise and initiative.

REFERENCES

[BERL72] Berliner, H. "Chess as Problem Solving: The Development of a Tactics Analyzer". Ph. D., Dissertation, Carnegie-Mellon University, 1972.

[BERL73] Berliner, H. "Some Necessary Conditions for a Master Chess Program". In Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973.

[BERL80] Berliner, H. "Some Observations on Problem Solving". Pittsburg, Pennsylvania: Carnegie-Mellon University Technical Report CMU-CS-80-113 1980.

[CARB79a] Carbonell, J. G. "Subjective Understanding: Computer Model of Belief Systems". Ph. D. Dissertation, Yale University (TR 150), January 1979.

[CARB79b] Carbonell, J. G. "Planning thru Adversity: The Counterplanning Process". In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, (1979): pp. 124-130.

[CHIE75] Chien, R. T. and Weissman S. "Planning and Execution in Incompletely Specified Environments". In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, (1975): pp. 169-174.

[deGR65] de Groot, A. D. "Thought and Choice in Chess". In Human Chess Skill, pp. 34-53. Edited by N. Charness. The Hague: Mouton, 1965.

[DOCK82a] Dockery, J. T. "Fuzzy Design of Military Information Systems". International Journal of Man-Machine Studies (1982) 16, 1 pp. 1-38.

[DOCK82b] Dockery, J. T. Personal Communications 1982.

[DUPU79] Dupuy, Col. T. N. Numbers, Predictions and War. New York: Bobbs-Merrill Co. Inc., 1979.

[ERNS69] Ernst, G. W. and Newell, A., GPS: A Case Study in Generality. New York: Academic Press, 1969.

[ESPO59] Esposito, V. West Point Atlas of American Wars. New York: Frederick A. Praeger Publishers, 1959.

[FIKE71] Fikes, R. E. and Nilsson, N. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving". Artificial Intelligence, 2 (1971): pp. 189-203.

[HARD75] Hardy, I. Chickamauga - The Last Victory, 20 September 1863. New York: Simulations Publication Inc., Copyright 1975.

[HARR74] Harris, L. R. "The Heuristic Search and the Game of Chess: A Study of Quiescence, Sacrifices and Plan Oriented Play". In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, (1974): pp. 334-339.

[HAYE79] Hayes-Roth, B. et. al. "Modeling Planning as an Incremental Opportunistic Process". In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, (1979): pp. 375-383.

[HAYE80] Hayes-Roth, B. "Projecting the Future for Situation Assessment and Planning". Santa Monica, California: Rand Corporation, Note N-1600-AF November 1980.

[LEIN68] Leinfellner, W. Theory of Human Evaluation and Decision Making. Lincoln, Nebraska: University of Nebraska Press, 1968.

[LESS77] Lesser, V. R., and Erman, L. D. "A Retrospective View of the HEARSAY-II Architecture". In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, (1977): pp. 790-800.

[LOWR78] Lowrance J., Grasper Language Reference Manual, Amherst, Massachusetts: University of Massachusetts Technical Report 78-20, December 1978.

[LOWR77] Lowrance, J. and Friedman, D. "Hendrix's Model for Simultaneous Actions and Continuous Processes: an Introduction and Implementation". International Journal of Man Machine Studies, 9 (1977): pp. 537-581.

[LUCE58] Luce, D. and Raiffa, H. Games and Decisions. New York: Wiley Press, 1958 Third printing.

[LUCE64] Luce, D. "Utility Theory". Mathematics and Social Sciences I, 1, (1964): pp. 55-71.

[McCA82] McCalla, G. I. and Reid, L. "Plan Creation, Plan Execution and Knowledge Acquisition in a Dynamic Microworld". International Journal of Man-Machine Studies 16, 1 (1982): pp. 89-112.

[McDE78] McDermott, D. "Planning and Acting". Cognitive Science 2 (1978): pp. 71-109.

[MINS75] Minsky, M. "A Framework for Representing Knowledge". In The Psychology of Computer Vision, pp. 211-277. Edited by P. H. Winston. New York: McGraw-Hill, 1975.

[MOST79] Mostow, D. and Hayes-Roth F. "Machine-Aided Heuristic Programming: A Paradigm for Knowledge Engineer". Santa Monica, California: Rand Corporation, Technical Report N-1007 1979.

[NEWE72] Newell, A., and Simon, H. A. Human Problem Solving. Engelwood Cliffs, New Jersey: Prentice-Hall Inc, 1972.

[NILS71] Nilsson, N. Problem Solving Methods in Artificial Intelligence. New York: McGraw-Hill Book Co., 1971.

[NILS80] Nilsson, N. Principles of Artificial Intelligence. Palo Alto, California: Tioga Publishing Co., 1980.

[PRAT81] Pratush, T. "Advanced Tactics: Reality and Games". Moves New York: Simulations Publications Inc., parts 1, 2, and 3; (54 December 1980/January 1981, 55 February 1981/March 1981, and 56 April 1981/May 1981).

[REIT79] Reitman, W. and Wilcox, B. "The Structure and Performance of the INTERIM.2 GO Program". In Proceedings Sixth International Joint Conference on Artificial Intelligence, (1979): pp. 711-719.

[RISS78] Rissland, E. "The Structure of Mathematical Knowledge". Cambridge, Massachusetts: Massachusetts Institute of Technology Artificial Intelligence Technical Report 472, 1978.

[RISS80] Rissland, E. "Example Generation". In Proceedings Third National Conference of the Canadian Society for Computational Studies of Intelligence, (1980).

[RISS82] Rissland, E.; Soloway, E.; Waisbrot, S. and Wall, R. "Constrained Example Generation - an Implementation". Amherst, Massachusetts: University of Massachusetts Technical Report in preparation.

[SACE74] Sacerdoti, E. D. "Planning in a Hierarchy of Abstraction Spaces". Artificial Intelligence, 5, 2 (1974): pp. 115-135.

[SACE77] Sacerdoti, E. D. A Structure for Plans and Behavior. New York: Elsevier North Holland, 1977.

[SACE79] Sacerdoti, E. D. "Problem Solving Tactics". In Proceedings Sixth International Joint Conference on Artificial Intelligence, (1979): pp 1077-1085.

[SAMU63] Samuel, A. L. "Some Studies in Machine Learning Using the Game of Checkers". In Computers and Thought, pp. 71-105. Edited by E. Feigenbaum. New York: McGraw-Hill 1963.

[SCHA77] Schank, R. C. and Abelson, R. P. Scripts, Plans, Goals and Understanding. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers, 1977.

[SCHA80] Schank, R. "Language and Memory". Cognitive Science 4, (1980): pp. 243-284.

[SHAN50] Shannon, C. E. "A Chess Playing Machine". Scientific American 182, #2, (February 1950): pp. 48-51.

[SIKL74] Siklossy, L. and Dreussi, J. "Simulation of Executing Robots in Uncertain Environments". Presented at the National Computer Conference, 1974, but left out of the Proceedings.

[SIMO80] Simonsen, R. Moves. New York: Simulations Publications Inc., 54 (December 1980/January 1981): p 1.

[SLAT77] Slate, D. J. and Atkin, L. R. "Chess 4.5 - The Northwestern University Chess Program". In Chess Skill in Man and Machine, pp. 82-118. Edited by P. Frey. New York: Springer Verlag, 1978.

[SPRO77] Sproul, R. F. "Decision Theory and Artificial Intelligence II: The Hungry Monkey". Cognitive Science 1 (1977): pp. 158-192.

[SUSS75] Sussman, G. R. A Computer Model of Skill Acquisition. New York: Elsevier Computer Science Library, 1975.

[USAR67] "Manuever Control". Baltimore, Maryland: U. S. Army AG Publication Center, FM-105 1967.

[USAR69] "Basic Staff Manual". Baltimore, Maryland: U. S. Army AG Publication Center, FM-101-5 1969.

[vonN53] von Neumann, J. and Morgenstern, O. Theory of Games and Economic Behavior. Princeton, N. J.: Princeton University Press, 1953, Third Edition.

[WALL81] Wall, R. S. "Scenarios in the Errand Running Domain", Amhesrt, Massachusetts: University of Massachusetts Technical Report 81-40 December 1981.

[WANG80] Wang, D. Personal Communications, 1980.

[WESS77] Wesson, R. "Air Traffic Control". Ph. D. Dissertation, University of Texas Austin 1977.

[WILE80] Wilensky, R. "Meta-Planning: Representing and Using Knowledge About Planning in Problem Solving and Natural Language Understanding". Berkeley, California: University of California Technical Report UCB/ERL M80/33 August 1980.

[WILK79] Wilkins, D. "Using Plans to Guide Search". Ph. D. Dissertation, Stanford University, 1979 STAN-CS-79-747.

ANNOTATED BIBLIOGRAPHY

This section contains a partial list of some of the background literature that aided the development of this thesis.

Brachman, R. "On the Epistemological Status of Semantic Networks." In Associative Memories, pp. 3-50. Edited by N. V. Findler. New York: Academic Press, 1979. A different graph-based language was proposed by Brachman called KLONE. Semantic networks had been used in numerous applications to represent associations among concepts, but many epistemological issues inherent in their use were ignored. KLONE formalized knowledge structuring links allowing the explicit construction of epistemological relationships.

Daniel, L. "Modifying Non-Linear Plans." Edinburgh, Scotland: University of Edinburgh DAI working paper 24 Dec 77. Project Networks - a form of PERT chart and the information it can provide is examined.

Davis, R. and King, J. "An Overview of Production Systems." In Machine Intelligence 8: Machine Representations of Knowledge, pp. 300-332. Edited by E. Elcock and D. Mitchie. Chichester: Ellis Horwood, 1977. Different models of ways to represent knowledge have been proposed to solve different classes of problems. All of the Expert Systems discussed in this work use production rules. These rules are

collected into a production system. Each rule is expressed as an IF...THEN statement, sometimes interpreted as Situation...Action in that if the situation exists then perform the action. The situation usually consists of a conditional expression over the facts in the world state. The action can be to add facts, delete facts, ask the user for more information, etc. Since application of a rule changes the fact base and the situation, different rules become applicable - engendering further changes, and so on. Eventually, it is hoped, the system would reach a stable state implying completion of the solution.

Feigenbaum, E. A. "Themes and Case Studies in Knowledge Engineering." Menlo Park, California: Stanford University Technical Report STAN-CS-77-621 1977. Work has been done in various complex domains to develop systems to aid humans in solving various tasks. This is a summary of the expert system paradigm and its manifestations.

Fikes, R. E., Hart, P. E. & Nilsson, N. J., "Some New Directions in Robot Problem Solving." In Machine Intelligence 7, pp. 405-430. Edited by B. Meltzer and D. Mitchie. Chichester: Ellis Horwood 1972. A summary of basic planning techniques, but more importantly, a speculation on planning domains that (at the time of writing) still needed to be explored. These included dynamic worlds, multiple outcome operators, multiple goals, multiple agents and hierarchical plans.

Goldstein, I. P. & Roberts, R. B. "NUDGE, A Knowledge-Based Scheduling Program." Cambridge, Massachusetts: Massachusetts Institute of Technology Artificial Intelligence Memo 405 February 1977. A system that uses frames to store knowledge about committees and project teams and can be used as a scheduling assistant to perform such tasks as making sure all members are free for a meeting.

Hendrix, G. "The Representation of Semantic Knowledge." In Speech Understanding Research, Edited by D. Walker. Palo Alto, California: SRI Inc. Final Technical Report 1976. Presentation of a knowledge descriptive representation: the semantic network. A semantic network is basically a graph structure to whose nodes and links various attributes have been added. Knowledge can be stored by correctly setting up the links and it can be recovered by traversing the graph.

Lenat, D. B., "Beings: Knowledge as Interacting Experts." In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tblisi, USSR (September 1975). An early discussion of the multiple-expert/knowledge-source paradigm.

Shortliffe, E. Computer Based Medical Consultations: MYCIN. New York: Elsevier, 1976. Shortliffe developed MYCIN which performed diagnoses of blood infections and suggested treatment. These and other systems can be considered problem solving systems. For example, in the MYCIN world, the problem is that the current world state does

not contain a diagnosis nor a suggested treatment and the goal state has both. Such a goal can be stated even though the details of the diagnosis and treatment are unknown. A solution involves the manipulation of facts, test results and medical knowledge until a valid diagnosis is created, satisfying the goal.

Stefik, M. J. "Planning with Constraints." Ph. D. Dissertation, Stanford University, 1980. MOLGEN, developed by Stefik, creates plans for molecular genetics experiments by using domain specific information to constrain a search of the operator space.

Tate, A., "Using Goal Structure to Direct Search in a Problem Solver" Ph. D. Dissertation, University of Edinburgh, 1975. Work was done examining interacting goals. The INTERPLAN system was used to detect when the system was faced with interacting goals, infer what the problem was, and redirect the operator search to get around the problem. Normally it would treat subgoals as independent processes unless a conflict was detected.

Walker, D. (Editor) Speech Understanding Research. Palo Alto, California: SRI Inc. Final Technical Report 1976. A complete summary of the work done by the people at SRI on the Defense Department's natural language research project.

APPENDIX

This appendix is an extraction of various portions of the CSG Programmer's Guide.

A. Block Diagram

This section outlines the workings of the display-symbol routine when displaying a scenario and the generate-scenario routine.

A.1. Generating a Scenario

If the generate command or the generate-scenario function is given the following processing is performed:

1. The scenario generation routine is entered [(build-scenario task situation name)].
2. A set of problems which will be used to interrogate the data base through CEG is created [(build-constraints goal situation)] by analysis of the goal and situation [(situation-analysis goal situation)], [(tactical-analysis behavior fr-str en-str)], [(enemy-behavioral-tactics en-str fr-str)].

3. Each problem is presented to the CEG system which searches the data base for similar experiences [(ceg-search cl-name)] and the results are collected.
4. Each remembered experience is modified to fit the current goal/situation exactly [(instantiate-scenario scen fc-list task situation *fr-str *en-str)].

A.1.1. Example Call History

The following is a partial call history of an invocation of (generate-scenario).

- I. build-scenario
 - A. build-constraints
 - i. situation-analysis
 - ii. enemy-behavioral-tactics
 - iii. tactical-analysis
 - B. bind-experience-base-to-example
 - C. choose-concept-space
 - D. set-up-search
 - E. ceg-search
 - i. ceg-exec-front
 - a. enemy-position-judge

- (i) compare-position
 - (a) comp-pos-en
 - (b) comp-pos-fr
 - (c) test-strength

- b. goal-judge
 - (i) goal-shape
 - (ii) goal-force

F. instantiate-scenario

Figure 28: Call History from Generating a Scenario

A.2. Displaying a Scenario

Assuming the user is at the Grinnell, the system will perform the following:

1. Break up the scenario object into its component parts (goal, situation, segment list, failure list and problem list).
2. Run the scenario synopsis routine [(scenario-synopsis s-list)].

3. Put the mapsheet in the background planes [(draw-map \$\$map)].
4. Enter the scenario display command loop.

When told to display a segment the system will:

1. Display information about the segment in the upper left corner of the overlay planes [(display-status seg)].
2. Calculate the screen coordinates of the force and goal locations as well as the front unit direction vectors.
3. Display the course of events ply by ply [(display-coe seg)] asking the user to hit the enter key on the cursor control box after each ply is displayed [(draw-coe wcoe frpos enpos goal frfr enfr gfr frstr enstr)].
4. Display the moves and combats described in the course of action [(draw-move fn pts)], [(draw-combat frcen encen)], [(draw-combat-result dscr frcen encen frfr enfr frstr enstr frinv eninv)].

5. Keep track of the position and status of the units and goals involved.

A.2.1. Example Call History

The following is a partial call history of an invocation of (display-symbol) on a scenario.

- I. display-scenario
 - A. scenario-synopsis
 - B. draw-map
 - C. display-segment
 - i display-status
 - ii display-coe
 - a. display-status
 - b. draw-goal
 - c. draw-force
 - d. draw-coe
 - (i) draw-move
 - (ii) move-result
 - (iii) draw-force
 - (iv) draw-combat
 - (v) draw-combat-result
 - (vi) combat-result
- II. grflush

Figure 29: Call History from Displaying a Scenario

B. Function Detail

This section discusses the major level functions of each subsystem (Scenario Generation, CEG Interface, and Graphics) in detail: what the functions and variables are, what the major data structures look like, what the system is supposed to be doing and why, and hints on changing and debugging.

B.1. The Scenario Generation Subsystem

This subsystem consists of the functions generate-scenario, build-scenario, build-constraints, situation-analysis, enemy-behavioral-tactics, tactical-analysis, and instantiate-scenario.

B.1.1. Generate-scenario

This routine is the entry point to the scenario generation subsystem. It gets the name the scenario is to have, the goal to be examined and the situation in which it is to be examined. It calls build-scenario and inserts the built scenario in the symbol table and the data file if so desired. The only data structures involved are the frames that make up the goal and the situation (lisp arrays). The scenario name is bound to the list of the goal, the situation, the list of instantiated experiences, the list of constraints of problems that were not resolved, and the constraints of problems that were

resolved; thus what is saved in the data file is the makeset of the unroll of the caddr of the value of the name (skipping the goal and situation, which should already be saved).

B.1.2. Build-scenario

This routine controls the top level scenario generation processing. This consists of first building the constraints by analysing the goal and situation. A series of problems is created which will then be presented to the data base through the CEG system. Theoretically, solving these problems is sufficient to recover the complete range of the future possibilities. Whether solution to all of these problems is necessary to cover the future is not known.

When all the problems have been tried, the experiences that were recovered, and the problems they are examples of solutions for, are given to the instantiate-scenario routine. The scenario name is then bound to the list of the goal, the situation, the list of experiences, the list of unsolved problems and the list of solved problems.

The only domain specific section of this routine is at the beginning of the repeat search through the problems, when the list of constraints is divided into the those that are simple binary constraints, a la a relational data base query (e. g., tactic: frontal-attack), and those that will require the full CEG judgement mechanisms (the goal and positional analysis constraints).

The list of problems returned by the constraint generation routine (build-constraints) is in the form of a list of names, each bound to a list of constraint names.

B.1.3. Build-constraints

This routine builds a list of problem names each bound to a list of constraint names. Each constraint is an expression that details a value to be matched to a similar value of an experience in the data base. The constraints are the semantic description of the forces involved in the situation, the goal expression to be achieved (from the specifics slot of the goal frame), the enemy behavior mode, the enemy tactic given that behavior, the random effects mode and the friendly tactic.

The semantic description of the involved forces is generated by the situation-analysis routine. It returns two lists of force descriptors, one for the friendly and one for the enemy forces. Each descriptor is a triple: <type name strength, where type is the semantic tag, name is the name of the unit/group, and strength is the current strength of the unit/group.

The goal constraint is taken directly from the goal frame.

The enemy behavior and enemy tactic constraints are generated by the routine enemy-behavioral-tactics, which analyses the situation and decides what the possible enemy behavior modes are and what tactics the enemy could use in this situation.

The random effects mode constraint is one of {helpful, neutral or detrimental}.

The friendly tactic constraint comes from the tactical-analysis routine. This routine generates all possible ways of achieving the goal given the forces available and the enemy forces in opposition.

The list of problems then consists of the situation analysis and goal constraints, which are the same for each problem, and all combinations of enemy behaviors and tactics, random effect modes and friendly tactics.

This routine is domain independent as long as you can fit your constraints into the above categories (in which case you may need new versions of the situation-analysis, enemy-behavioral-tactics, and tactical-analysis routines), otherwise you would have to create your own.

B.1.4. Situation-analysis

This routine builds the list of the lists of enemy and friendly force descriptors. This is done by calculating where on the map each force and goal is, and then checking what positional relationship there is among them. The enemy forces are analyzed by describe-enemy-position and the friendly forces by describe-friendly-position.

B.1.4.1. Describe-enemy-position

For each enemy force there are four possible descriptor types: on, between, near or outside. This routine tests an enemy force by first checking if it is on any of the locations of the goals to be achieved. If one of the goals is to destroy this enemy force then obviously it is on a goal location. The descriptor is then on or the list (on n) if there is more than one goal and the force is on goal n.

If the force is not on a goal, then it is tested to see if it is between a friendly force and a goal, where between is defined to mean within a circle drawn through the goal and friendly force with an origin at the midpoint between the goal and the friendly force. If so, then the descriptor type between is returned or the list (between gn fn) if there is more than one goal (gn) or friendly force (fn) involved.

The next test is to see if the enemy force is within 7 hexes of a goal. If so, then the descriptor type is near or the list (near n) if there is more than one goal and the force is near goal n.

If the force fails all of the above tests then it is described as outside.

B.1.4.2. Describe-friendly-position

For each friendly force there are four possible descriptor types: on, next, near or outside. This routine first checks to see if the unit is on a goal location. If the force is to be protected then it is obviously on a goal. The descriptor type is then on or the list (on n) if there is more than one goal and the force is on goal n.

If the force is not on a goal, then it is tested to see if it is adjacent to an enemy force, since by the rules such a force is committed to immediate combat. If so, the descriptor type is next or the list (next en) if there is more than one enemy force and the friendly force is next to enemy force en.

The next test is to see if the friendly force is within 7 hexes of a goal. If so, the descriptor type is near or the list (near n) if there is more than one goal and the friendly force is nearest goal n.

Finally, if the friendly force is not on, next or near it is automatically classified as outside.

B.1.5. Enemy-behavioral-tactics

This routine examines each possible enemy behavior mode and for each one calls the routine tactical-analysis to generate the possible tactics the enemy could perform given the forces involved.

It returns a list of the enemy behavior/enemy tactic constraint pairs.

B.1.6. Tactical-analysis

This routine takes a behavior mode a list of friendly forces and a list of enemy forces (who is enemy and who is friendly is left to the calling routine) and returns a list of all possible tactics that the friendly forces could perform.

If the behavior mode is aggressive then the possible tactics are determined by the number of friendly and enemy forces involved. If there are no enemy forces, then the only tactic is column-advance. If there are no friendly forces then the only tactic is sit. Otherwise, the tactic frontal-attack must be considered. If there is more than one friendly force, then encirclement is also considered. Finally, if there is more than one enemy force involved the tactic flank-attack is

considered.

If the behavior mode is passive then the possible tactics are also determined by the number of friendly and enemy forces involved. If there are no enemy forces, then again the tactic column-advance is the only one suggested. If there are no friendly forces then sit is the only tactic. Otherwise, the tactic fluid-defense is suggested. Finally, if there is more than one friendly force the tactic line-defense is also suggested.

B.1.7. Instantiate-scenario

This routine calls recursively instant-scen and inst-scen which do the following work to modify remembered experiences to fit the current situation exactly (instantiation).

A copy of the experience to be instantiated is made. In the copy, the values of the slots tactic, random-effects, enemy-tactic and enemy-bahavior are replaced with the values specified by the constraint. The reason there may be a difference is if the experience is a model experience in which case it may have more than one value for the given slot.

The value in the name slot is replaced with the name of the remembered experience. The friendly and enemy strength slots are replaced with the descriptor lists from situation-analysis. The friendly and enemy names are replaced with those from the situation frame as are the friendly and enemy location slots.

The goal slot is replaced with the value from the specifics slot of the goal frame, and the destination slot with the goal-locations of the goal expression.

Finally, the coe (course of events) slot in the remembered experience is examined. If the first element of the value is prog then the coe slot in the copy is set to the eval of the remembered experience coe. Otherwise, it is left as it was remembered.

B.2. The CEG Interface Subsystem

This subsystem consists of the functions bind-experience-base-to-example, choose-concept-space, set-up-search, ceg-search, ceg-exec-front, enemy-position-judge, compare-position, comp-pos-en, comp-pos-fr, test-strength, goal-judge, goal-shape and goal-force. It also uses the Constrained Example Generation system.

B.2.1. Bind-experience-base-to-example

The normal form of the experiences is as a simple array. The Constrained Example Generation system, however, expects the examples in the database it is searching to be arrays with one of the slots being the value of the example. This array of an array is too much to carry around and store all the time, so we defer actually building the example database that CEG will use until as actual generation is in progress, and then, only bind those experiences that have been chosen by choose-concept-space.

Choose-concept-space returns the list of potentially applicable experiences. Bind-experience-base-to-example takes each member of this list and creates a dummy shell array that looks like the array expected by CEG as an example - the slots have the correct values. It binds each of those arrays to an atom that is the name of the experience with "x:" concatenated on the front.

B.2.2. Choose-concept-space

There are currently 6 constraints used to select experiences from the database. Two of these, goal and enemy-position, are sent to CEG which does a complete analysis and judgement. The other four, tactic, enemy-behavior, enemy-tactic and random-effects are not semantic and are used as simple atomic labels, e. g., tactic = frontal-attack, or random-effects = detrimental.

Thus a structure has been created that facilitates the relational database nature of these constraints. (The structure was created using create-concept-space, which takes a list of experience names and adds them and their attributes to the current structure.) The function choose-concept-space takes a list of attribute-name/value pairs and uses them to select the list of potentially applicable experiences.

The structure \$\$concept-space is an attribute list of attribute lists of ordered sets. The top level attribute list is the list of constraints (random-effects, tactic, etc), each with a value that is in turn an attribute list of the possible values that that constraint can have (e. g., for random-effects, helpful, neutral and detrimental). Finally, for the value of each of the names of these low level attributes we have the ordered set of the names of all the experiences that have that value of that constraint (e. g., the list of all experiences that show random-effects = detrimental).

The routine choose-concept-space then selects each correct attribute value experience list as specified by its given list of pairs and does an ordered set intersection on them to get the list of experiences that have all the specified values of the constraints. A working version of \$\$example-list (the list used by CEG to see which example should be examined next) is then created.

This pre-CEG selection process speeds up the normal CEG search substantially.

B.2.3. Set-up-search

This routine merely sets two of the Constrained Example Generation System policy values, the ground level policy and the modification threshold. Two parameters are passed to it, the policy and the threshold to set. Normally, the policy is a single epistemological type: su, ref, or model, as defined by CEG.

B.2.4. Ceg-search

This routine calls for constrained example generation repeatedly until all possible relevant experiences have been retrieved. It does this by manipulation of the constrained example generation policy parameters.

B.2.5. Ceg-exec-front

This routine, part of the constrained example generation system itself, is the entry point through which the constrained scenario generation system calls constrained example generation.

B.2.6. Enemy-position-judge

This routine is called by the constrained example generation system to determine if a potential experience is relevant. It calls compare-position which does the actual comparison of force descriptors.

B.2.6.1. Compare-position

This routine does first a simple count to see if the current situation and the retrieved experience have the same number of involved forces. It then in turn calls comp-pos-en and comp-pos-fr which do the semantic comparisons.

B.2.7. Goal-judge

This routine does the comparison between the problem goal statement and the experience problem statement. It calls goal-shape, which tests the AND/OR keyword structure of the two goal statements, and goal-force, which tests the forces involved to see which forces have been assigned to the same task.

B.3. The Graphics Subsystem

The main routines of the Graphics subsystem are display-scenario, scenario-synopsis, display-segment, display-coe and draw-coe.

B.3.1. Display-scenario

This is the top level entry point to the graphical display of the assembled scenario set. It checks to see if the GMR-27 is allocated and displays the game board if not already present.

B.3.2. Scenario-synopsis

This routine shows the user at the terminal the ways in which the scenario segments can be organized. It allows the user to see how many examples there are of each possible world category.

B.3.3. Display-segment

This routine is called with an experience to be displayed. It sets up the initial translations of course of events descriptions to screen coordinates.

B.3.4. Display-coe

This routine loops through the course of events shown in the experience, prompting the user to press the cursor Enter key after each ply is shown. The user may enter Home-Enter to abort the rest of the display of this course of events.

B.3.5. Draw-coe

This is the routine that actually puts the course of events on the screen. It makes the translations of course of events keywords to graphics coordinates.