A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

THESIS

Garrison H. Flemings
Major, USAF

AFIT/GST/ENS/92M-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GST/ENS/92M-03

A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

THESIS

Garrison H. Flemings
Major, USAF

AFIT/GST/ENS/92M-03

THESIS APPROVAL

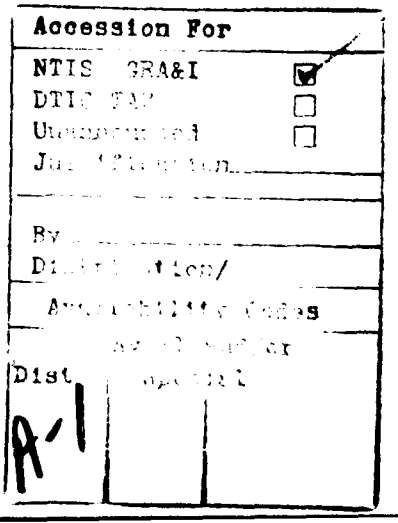
STUDENT: Major Garrison H. Flemings

CLASS: GST-92M

THESIS TITLE: A Decision-Theoretic Approach to Recommending Action in the Air-to-Ground Aircraft of the Future

DEFENSE DATE: 2 March 1992

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Major Bruce W. Morlan/ENS	Bruce W. Morlan
Reader	Colonel Thomas F. Schuppe/ENS	Thomas F. Schuppe



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1992	Master's Thesis

4. TITLE AND SUBTITLE
A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

5. FUNDING NUMBERS

6. AUTHOR(S)
Garrison H. Flemings, Major, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology,
WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GST/ENS/92M-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint Cockpit Office
Systems Development Branch
Wright Laboratory
Wright-Patterson AFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Designers of air-to-ground fighters of the future should consider including decision support systems to relieve pilots of some of the present workload. Those decision support systems should use utility theory. This work reviews two techniques of utility theory (decision trees and influence diagrams) and a new technique due to Morlan (probability ratio nets). The work includes a proposed structure for action recommendation systems in air-to-ground fighters of the future. The work describes an implementation of probability ratio nets. The work demonstrates the value of decision theory to air-to-ground fighters of the future by way of solving an example problem and several variation of it.

14. SUBJECT TERMS
Utility Theory; Pattern Recognition; Game Theory; Statistical Decision Theory

15. NUMBER OF PAGES
182

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT/GST/ENS/92M-03

A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science (Operations Research)

Garrison H. Flemings, B.S., M.S.
Major, USAF

March, 1992

Acknowledgments

My family deserves much credit for their help during this thesis effort. They motivated me to pay attention to "life outside of the thesis" and hence improved both my life and work. My relationships with Kathy, my wife, and with our two children, Jennifer and Richard, are the core of my life.

In developing Example 2 (the "T/APC/SAM Problem") in Chapter 5, I hoped to develop an example with some ring of truth to it. I got help from two fighter pilots: Maj John Stieven and Matt Gaebler. The extent to which the example represents the battlefield at the Forward Edge of the Battle Area is due entirely to their help. The extent to which the example fails to represent the battlefield and the vehicles involved is due entirely to me.

The input of Col Thomas F. Schuppe to this work was pivotal. As I planned the project, it was Col Schuppe's input that prevented attempting to complete some untenable avenues. Col Schuppe's feedback on drafts of this document was insightful and balanced. I deeply appreciate Col Schuppe's contribution in time and judgment.

It is clear that this effort stands heavily on the work, experience, and judgment of Maj Bruce Morlan. His enthusiasm and his knowledge of the fields of utility theory and of Bayesian statistics are inspirations. The technique of probability ratio nets he developed and had me implement has potential for efficient implementations of systems based on utility theory.

Garrison H. Flemings

Table of Contents

List of Figures

List of Tables

Abstract

Designers of air-to-ground fighters of the future should consider including decision support systems to relieve pilots of some of the present workload. Those decision support systems should use utility theory. This work reviews two techniques of utility theory (decision trees and influence diagrams) and a new technique due to Morlan (probability ratio nets). The work includes a proposed structure for action recommendation systems in air-to-ground fighters of the future. The work describes an implementation of probability ratio nets. The work demonstrates the value of decision theory to air-to-ground fighters of the future by way of solving an example problem and several variations of it.

A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

I. Introduction

1.1 Background of the Problem

The United States operates and is developing many systems to collect information about the battlefield. For many years, the United States has operated and produced the Airborne Warning and Control System (AWACS). The AWACS is an aircraft with a rotating antenna mounted above the fuselage which combat forces use to observe the battlefield environment and to direct forces. Today, the Air Force and Army are jointly developing the Joint Surveillance Target Attack Radar System (Joint STARS), a follow-on aircraft based on a phased array radar (Aerospace Amer, Feb 90, AF Magazine, Jun 91). In addition, the United States has operated satellite systems for many years. Those systems improve reconnaissance capabilities, too. However, there are hints that some systems still do not perform adequately.

The modern battlefield is confusing. On 3 July 1988, a U.S. cruiser, the Vincennes, engaged some small Iranian gunboats in battle. This action was the first time the crew had been in combat (NY Times, 20 Aug 88:5). Coincidently, the engagement took place underneath the planned flight path of a civilian airliner which took off during the action from an Iranian airfield used by both military and civilian aircraft. The crew of the Vincennes erroneously assessed the airliner

as being an Iranian F-14 making an attack and the crew shot the airliner down. Similarly, early in the Desert Storm operation, a Maverick air-to-ground missile launched from a U.S. A-10 ground attack aircraft diverted in flight from the target tank to a U.S. Marine light armored vehicle positioned on the flight path to the target (Dodging Friendly Fire). Upon impact, the missile proved its lethality—seven U.S. Marines died. These two examples demonstrate the confusion of the modern battlefield.

On tomorrow's battlefield, there is hope for making fewer such decisions. However, today's pilots report feeling bombarded by the quantity of data they get in their cockpits. One active program to address this issue is the Pilot Associate Program (contract number F33615-86-C-3802). The following is from a report prepared by McDonnell Aircraft Company for that program.

> Today's combat pilots are provided with the world's best equipped aircraft and are asked to maximize the combined performance of pilot and machine under severe conditions. To do this, they must continually assess their aircraft's capabilities and the situations surrounding them. They are often forced to make split-second life or death decisions and execute their plans with conflicting or incomplete information. Tomorrow's pilots will face even bigger challenges against more capable threats while flying aircraft having even more airframe and avionics capability. Designers of such aircraft must provide the on-board support systems necessary to enable the pilot to do those things he can do best—fly and fight—instead of performing more routine functions, such as avionics systems control. (Final Report 91:1-1)

The Joint Cockpit Office, part of the Cockpit Integration Directorate of the Wright Laboratory at Wright-Patterson AFB, Ohio, is using an innovative idea in designing the cockpit for the Multi-Role Fighter, a weapon for tomorrow's battlefield. The office is designing that cockpit unconstrained by other parts of the aircraft design team (fuselage, engine, flight control, etc.). Their hope is that because they are using this concept of design, their cockpit design will better meet the needs of pilots than

any existing cockpit. Should they achieve such a result, other members of the aircraft design team may be much more willing to accept constraints from them.

In this approach to design, the office is starting work with basic questions like, "Can achievable technology improve on the current presentation of information to the pilot? Can we reduce pilot workload by presenting processed information on the displays?" Better cockpit designs have lower pilot workload, all other factors being equal.

There is great potential in cutting pilot workload by better processing the information provided to pilots. Indeed, the purpose of the Pilot's Associate program is "to create a layer of intelligent avionics" (Final Report 91:1-1) to help the pilot with routine monitoring functions and critical combat-related tasks. If machines can propose decisions the pilots trust, the combat system might be more effective.

1.2 Purpose of the Research

This investigation defines an action recommendation system as a hardware capability to give future fighters the capability of processing information from a sensor suite and from rules of engagement for the purpose of recommending to the pilot which of several actions is most appropriate. This effort proposes to use utility theory (from the field of Decision Analysis) as a technique to offer decisions to pilots based on information available at decision-making time. Utility theory is appropriate because it is so powerful a way to combine information known about the state of the battlefield with information about general goals to recommend a course of action.

In particular, this project will consider aircraft used to attack ground vehicles. This limit is a convenience for this thesis effort; the work can extend to many decisions made in the aircraft system and can further extend to many other environments.

Within utility theory, this project will assess the technique of probability ratio nets developed by Morlan (Morlan 91). The technique is new to the literature and

may provide an efficient means of capturing the decisions the aircraft system must make.

1.3 Assumptions

This thesis effort involves these assumptions.

- Some element in the aircraft system will identify a potential target as an item of interest and list all possible target identities. Some element in the aircraft system will deliver that information as inputs to the action recommendation system. Some of these inputs might come from the pilot or all inputs might come from some machine element of the aircraft system.
- Some element in the aircraft system will provide the action recommendation system an exhaustive list of possible actions for the aircraft in response to the potential target. An example of a possible action list would be: "attack", "avoid", "gather further information", and "ignore".
- Some element in the aircraft system will accept the output from the action recommendation system. For example, the output might go to a moving map display responsible for indicating likely target type and proposed action in response.
- Sensor data on the item of interest will be available to the action recommendation system. Sensors might be on the aircraft, like a radar mapping device or an infrared camera. Other sensors might be separate from the aircraft and their data might be sent to the aircraft by a data link. Data produced from satellite imagery is an example of external data.
- All sensor information will be complete. Thus, this project's system need not address problems created by incomplete information.

- Each measure of the item of interest will be discrete. This simplifying assumption keeps the project size appropriate to a thesis-level effort. After completion of this project, follow-on research could work with continuous data.
- For this project, computation speed is not an issue.

1.4 Document Overview

Chapter 2 reviews the pertinent literature, pointing out strengths and weaknesses of various techniques of decision analysis. Chapter 3 demonstrates three techniques (decision trees, influence diagrams, and probability ratio nets) on a very simple example problem with the goal of introducing the reader to each of the three techniques. Chapter 4 proposes a logical structure for the class of problems faced by designers of decision-support equipment for the air-to-ground fighter of the future. Information in that chapter will be of value to system designers because it will help them orient their thinking about their designs; the information will be of value to pilots who evaluate designs of new aircraft because it will help them understand these new systems and evaluate the potential of this valuable application. Chapter 5 discusses implementation of the ideas in Chapter 4 and presents solutions to two problems. The second problem, more complex than the first, includes several variations. The chapter includes solutions to each variation and explanation of the significance of each solution; like Chapter 4, this example will help pilots who evaluate new aircraft designs and will help the reader understand the value of decision analysis in fighters of the future. Chapter 6 presents two conclusions and some recommendations for future research.

II. Literature Review

Two bodies of the literature are relevant to this project: work done with systems decision-making and techniques of decision analysis. This chapter reviews each.

2.1 Research Into Decision-Making

2.1.1 Biological Motivation in Research Much research in this field is biologically motivated. These researchers note the strong performance advantages biological systems have over current digital computers. For example, a comparatively simple organism, the pigeon, can do some very basic things no machine can now do.

> In experiments at the University of Iowa, eight trained pigeons were shown photographs of different people displaying emotions of happiness, anger, surprise, and disgust. The birds learned to distinguish between these expressions. (NY Times, 2 May 89)

Similarly, work in 1942 of the noted psychologist B. F. Skinner led to a successful demonstration of a proposed missile guidance system using trained pigeons to increase missile accuracy (IEEE Spectrum, Aug 87). The prime interest of this line of research is to describe the mechanisms of biologic systems. These researchers expect that better understanding of the functioning of biologic systems will suggest better ways to build machines for the same functions.

2.1.2 Building Decision Machines Another large element of research in this field has to do with building decision-making machines. For example, McDonnell Aircraft Company worked on systems for the air-to-ground fighter of the future under the Pilot's Associate Program. In a simulator, they showed the results of their work. The flight

> was a single ship air-to-ground battlefield interdiction sort [sortie] in a post-1995 Central European combat environment. ... Both targets were armored columns 20 nautical miles (MN) [NM] apart advancing on a network of roads approximately 60 nm from the FEBA [Forward Edge of the Battle Area]. (Final Report 91:2-4)

Because McDonnell Aircraft Company assumed a sortie attacking so far from the FEBA, they could assume there would be no friendly vehicles there. Because their assumed sortie spent so much time over enemy territory, their system needed to handle unexpected enemy defenses. They spent a lot of time developing the capability to change the attack route and to react to new threats, but their report has little mention of any time they spent characterizing potential targets of opportunity. These differences make the work of McDonnell Aircraft Company significantly different from the work of this thesis.

Bajcsy expresses a thought of many researchers working with pattern recognition.

> Most past and present work in machine perception has involved extensive static analysis of passively sampled data. However, it should be axiomatic that perception is not passive, but active. Perceptual activity is exploratory, probing, searching; percepts do not simply fall onto sensors as rain falls onto ground. (Bajcsy 88:279)

Zelnio, of the Target Recognition Branch in the Wright Laboratory, is working on a related technique called "model-based vision (MBV)" (Zelnio 91). The MBV paradigm uses a two-stage analysis process, first estimating potential locations in the automatic target recognizer solution space for the target. Those first estimates select which of several finely tuned filters the model will use for detailed analysis and identification. An especially attractive possible use of MBV might be with a FLASER, which "combines properties of both FLIR [forward looking infrared] and laser" (Zelnio 91:para 5). In this system, the infrared image might quickly identify areas of interest for the slower laser sensor to image, creating three dimensional

data on the area. The decision-making apparatus might make use of both sensors in identifying the target. Zelnio's work includes little work on the algorithms to use in deciding what to do given the developed information. The Advanced Target Recognition Working Group Technology Committee, sponsored by the Defense Advanced Research Projects Agency, considered the approach of model-based vision, describing it as "immature with respect to understanding, real-time implementation, and testing" (ATRWG Tech Comm:6). The same committee identifies the following factors in favor of model-based vision: the promise of "generalization and graceful degradation, with error increasing as obscuration becomes more pronounced" (ATRWG Tech Comm:7) and "the potential of adaptive segmentation to improve performance" (ATRWG Tech Comm:7).

Active work at the Air Force Institute of Technology in this area includes work in pattern recognition. For example, the work of Tarr (Tarr 91) showed new techniques for systems to identify portions of an image which are of particular interest, a process called "segmentation". However, there is much work remaining on this important part of the problem. The work of Singstock (Singstock 91) was to identify the objects in pictures after segmentation. He demonstrated computation of many features from infrared images and he demonstrated using these features to characterize target identity. The work of Tarr and Singstock is distinct from the work of this thesis in that each devoted their efforts to characterization of targets, a different problem than deciding what action to take.

2.2 Survey of Decision Analysis

A branch of the operations research field interested in making decisions is decision analysis. Of all the branches of the field, only this one combines information known about the operating environment with values for the outcome of actions to recommend which of several actions is most appropriate. Decision analysis is itself

a rich field and the literature covers many techniques. This review covers three techniques: decision trees, influence diagrams, and probability ratio nets.

2.2.1 Decision Trees Decision trees are, perhaps, the most commonly taught technique of decision analysis. They are directed graphs containing two kinds of nodes (Quinlan 90). "Decision nodes" depict all options a decision maker may choose at decision time. "Chance nodes" depict possible events which are outside the control of the decision maker. Lines between nodes are "branches"; they document relationships between nodes. Chapter 3 contains an example of a decision tree applied to a simple problem.

Though this technique has become a standard of decision analysis, there are problems. One problem is building the trees in the real world. Quinlan observes, "The basic algorithm for constructing decision trees ignores complexities that arise in real-world classification tasks" (Quinlan 90:342).

Another problem is the evidence that these trees do not model real decision making, a problem faced equally by all three techniques mentioned here. Schoemaker surveyed research into real decision making by individuals, and concluded real decision makers use different techniques—"The research reviewed in this article suggests that at the individual level EU [expected utility] maximization is more the exception than the rule, at least for the type of decision tasks examined" (Schoemaker 82:552). Howard also surveyed research into decision making, concluding, "man is considerably less skilled in decision-making than expected" (Howard 83:14).

A further problem is with implementation: some computer codes implementing decision trees have used large amounts of limited computer resources. White makes note of this problem and suggests another technique.

> We further remark that decision trees can have enormous storage requirements. Influence diagrams are a potentially more parsimonious graphical representation of knowledge necessary to determine a most preferred policy. (White 90:359)

2.2.2 Influence Diagrams Like decision trees, influence diagrams are directed graphs. Influence diagrams have three types of nodes: "chance nodes", "decision nodes", and "value nodes" (Howard 89). Chance nodes represent events over which the decision maker has no control. Decision nodes represent decision events. Value nodes represent the output of the tree. Arrows between nodes depict "relevance". Thus, an arrow between two chance nodes documents that the first event is relevant to the distribution of the second. Said another way, the distribution of the second chance node is conditionally dependent on the first chance node. Similarly, arrows into a decision node represent information expected to be known to the decision maker at decision time.

Once having formed an influence diagram, the decision analyst methodically uses one of several identified tools to transform the diagram through a series of steps (Tatman and Shachter 90). Eventually, the analyst transforms the net into an equivalent net with information presented as needed by the decision maker. Chapter 3 contains an example of an influence diagram applied to a simple problem.

Influence diagrams give the analyst a major advantage in interviewing the decision maker and collecting needed data. They clearly depict relevance of each part of a problem to the rest of the problem. In the words of Howard,

> The influence diagram is a major aid in this transformation [from people's heads to computer representation] because it crosses the border between the graphic view of relationships that is very convenient for human beings and the explicit equations and numbers that are the province of present computers. (Howard 83:14)

On the other hand, influence diagrams have problems, too. From Tatman and Shachter, "A shortcoming of the traditional influence diagram is that the separable nature, if any, of a value function is not revealed in the graphical structure and thus cannot be exploited." (Tatman and Shachter 90:365)

Howard has proposed a special case of influence diagrams he calls "knowledge maps" (Howard 89). Knowledge maps are influence diagrams with only chance nodes allowed and simplify some communication with clients. Otherwise, they are so similar to influence diagrams that this paper makes no further mention of the technique.

2.2.3 Probability Ratio Nets Morlan's probability ratio nets (PRN's) (Morlan 91), share with influence diagrams the attribute of being directed graphs. They also share having a process of using one of several identified tools to transform the nets into more useful representations. Nodes in PRN's are events, much like nodes in influence diagrams. However, the arcs between nodes represent the relative probability of the two events. Chapter 3 contains an example of a PRN applied to a simple problem.

A prime value of PRN's, in Morlan's opinion (Morlan), is the ability to handle differing information for various hypotheses. For example, he pointed to a problem of distinguishing engine types. For a diesel engine, the presence or quantity of smoke coming from the exhaust might be relevant to the analyst. For a gasoline engine, the analyst would very possibly not collect data on smoke for lack of relevance. He feels handling these situations is easier in PRN's.

Further, says Morlan (Morlan), PRN's are easier to add information to. That is, once an analyst has completed a net, adding information to a PRN is usually a quick and straightforward operation. Adding the same information to an influence diagram might require quantifying a large number of distributions relative to variables already in the net.

2.3 Conclusion

The field of pattern recognition is approaching the point of needing to make recommendations on actions to take based on the information known about the environment. The field of decision analysis has much to offer, including at least three ways to represent knowledge and process it into decisions. The purpose of this thesis is to advance this intersection of disciplines.

III. Three Approaches to Decision Analysis

This chapter details three methods of doing the mathematics of decision analysis: decision trees, influence diagrams, and probability ratio nets. The first section of the chapter poses a problem to solve (the "A–Type/B–Type Problem"), worded from the standpoint of advising a pilot on action in response to finding a potential target. The second section is a brief introduction to Bayes' Theorem, a basis for each technique of decision analysis introduced here. The remaining sections of the chapter solve the A–type/B–type Problem, each section using one of three methods of decision analysis. The section on decision trees contains little detail of the method, because there has been so much discussion of decision trees in the literature for so many years. The section on influence diagrams contains more detail to help introduce readers who have little exposure to the technique. The section on probability ratio nets discusses the technique in detail, because no other material on the technique is available.

3.1 Example 1: The A–Type/B–Type Problem with Two Questions

The problem presented here represents, in highly simplified form, the class of problem this thesis effort addresses.

3.1.1 Situation At a time of interest, an enemy has just launched a surface-to-surface missile. There are two variants of the launcher: the "A–type" and the "B–type". Intelligence reports suggest that near the launch location the enemy employs half again as many of the A–type missile launchers compared to the B–type launcher.

The launcher types differ in reaction time and defensive armament. The B–type launcher prepares itself for missile launch faster. Thus, at the time of interest, the B–type launcher is twice as likely to be ready for launch as to be not ready. At

the time of interest, the situation for the A-type launcher is reversed—it is twice as likely to be not ready for launch as to be ready. In a second difference between the two types of launcher, the B-type launcher has added defensive equipment which increases the risk to a fighter making a strafing pass.

Both launcher types emit an identifiable radio signal when the launcher is ready for launch.

The air order of battle places values on attack options. Attacking A-type launchers with a strafing pass is twice as desirable as using a missile. Attacking B-type launchers with a missile is twice as desirable as a strafing pass. A B-type launcher is four times as desirable to destroy as an A-type. Either type launcher is twice as desirable to destroy if it is ready for launch.

An air-to-ground fighter of the future is patrolling this area of the battlefield and seeks to destroy the launcher which has just fired. The fighter's sensor suite has detected a launcher of unknown type and determined that the launcher is not emitting the "ready for launch" signal.

3.1.2 Question Number 1: What Identity to Display? Assuming the fighter has a moving map display which must display a symbol for either "A-type" or "B-type", a problem facing the designer of the map display is to decide which symbol to use.

3.1.3 Question Number 2: What Action to Recommend? A further problem is to recommend which armament to use.

3.1.4 Preliminaries to Solution: Utilities When the analyst gets to the point of using the information given on target value, this problem statement forces an assumption. The analyst needs the assumption because all given utility information is relative; no statement assigns a utility value to any target/ordnance combination. However, making such an assumption is not troubling, because any assumption is

reasonable, so long as the analyst arrives at correct relative values for all other utilities.

Without loss of generality, this discussion assumes that the lowest value target/ordnance combination is worth "1". All other utilities follow; Table 3.1 presents results.

	A-type		B-type	
Ordnance	Not Ready	Ready	Not Ready	Ready
Missile	1	2	8	16
Strafe	2	4	4	8

Table 3.1. Utility Values Computed After Assuming the Lowest Utility is "1"

To some, these utility values will seem artificial and distinct from real-world practice. However, often there will be data from which decision-makers or their staffs can develop such values. These numbers might capture relationships between target value and attacker value. These numbers might capture tradeoffs between attacking these targets or attacking other targets. Other research might reflect more on developing these numbers; this thesis assumes that these numbers exist.

3.2 Bayes' Theorem

There are four elements to Bayes' Theorem. This section introduces each in the context of the example above. Then, it presents Bayes' Theorem and applies it to the problem.

3.2.1 Prior Probability The *prior probability* , $P(E)$, is an initial estimate of events of interest. In the A-type/B-type problem, the events of interest are the events of observing launchers of the two identities. The prior probability is the initial estimate of the distribution of launcher types, as shown in Table 3.2.

Event, E	Probability of Event, $P(E)$
A–type	3/5
B–type	2/5

Table 3.2. Prior Distribution, $P(E)$, for the A–Type/B–Type Problem

3.2.2 Conditional Probability The *conditional probability* is a set of probability distributions, one for each possible observation. In the A–type/B–type problem, the observations are the indications of readiness to fire—"Ready" and "Not Ready". Together, this set of distributions specifies the probability of each observation given each event, $P(O|E)$. Table 3.3 presents the conditional probability information for the A–type/B–type problem.

	Events	
Observation, O	A–type	B–type
Ready	1/3	2/3
Not Ready	2/3	1/3

NOTE: Each column is a probability distribution.

Table 3.3. Conditional Probability Information, $P(O|E)$ for the A–Type/B–Type Problem

3.2.3 Preposterior Probability The *preposterior probability* is the probability of any observation, $P(O)$. It is a function of the prior probability and the conditional probability by the formula

$$P(O) = \sum_i P(O|E_i) \cdot P(E_i)$$

where i is an index over all possible evidence states. In the A– type/B–type problem, the evidence states are "A–type" and "B–type". Thus, for that problem,

$P($ "Ready" $) = P($ "Ready" $|$ A–type $) \cdot P($ A–type $)+$

$$
\begin{aligned}
& P(\text{ "Ready" } \mid \text{ B-type }) \cdot P(\text{ B-type }) \\
&= 1/3 \cdot 3/5 + 2/3 \cdot 2/5 \\
&= 7/15
\end{aligned}
$$

$P($ "Not Ready" $)$ is computed similarly. Table 3.4 presents the result.

Observation, O	Probability of Observation, $P(O)$
Ready	7/15
Not Ready	8/15

Table 3.4. Preposterior Distribution, $P(O)$ for the A–Type/B–Type Problem

3.2.4 Posterior Probability and Bayes' Theorem The *posterior probability* is a set of probability distributions, one for each observation. Together, they specify the probability of each event, given each observation, $P(E|O)$. Bayes' Theorem states that posterior probability is a function of prior, conditional, and preposterior probabilities by the formula

$$P(E|O) = P(E) \cdot \frac{P(O|E)}{P(O)}$$

Table 3.5 presents the posterior probability information for the A–Type/B–Type problem, computed using Bayes' Theorem.

3.3 Decision Tree Solution

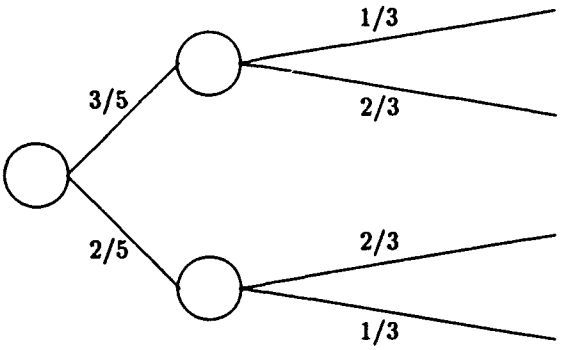
3.3.1 Development of Solution Decision trees are a now-classic approach to solving problems of this nature. Figure 3.1 shows the first tree resulting from this analysis. Raiffa discusses development of trees in detail (Raiffa 68).

Figure 3.1 depicts the probability information given in the problem, but it does not suggest symbology for the moving map display. An equivalent and more

Event, E	Observations, O "Ready"	"Not Ready"
A–type	3/7	3/4
B–type	4/7	1/4

NOTE: Each column is a probability distribution.

Table 3.5. Posterior Probability Information, $P(E|O)$ for the A–Type/B–Type Problem



Launcher	Ready to Launch?
A–type	Yes
B–type	No

NOTATION: Labels at the bottom identify arc meaning by position. Thus, on the right, the upper arc of each pair corresponds to answering the question "Ready to Launch?" with a "Yes" and the lower arc corresponds to a "No". Numbers on arcs are the probabilities associated with the arc.

Figure 3.1. Initial Decision Tree

useful representation results from a technique Raiffa calls "flipping the probability tree" (Raiffa 68:17). Figure 3.2 presents the result. From Figure 3.2, the display designer can conclude from probability alone that the appropriate symbol should be for the A–type launcher. Only the lower half of the tree applies, since the launcher in question has no indication of being ready for launch. In the lower half of the tree, the probability is 75 percent that the launcher is of A–type. If the display designer seeks to display the highest-probability symbol, the display should show A–type. However, Figure 3.2 is still incomplete.

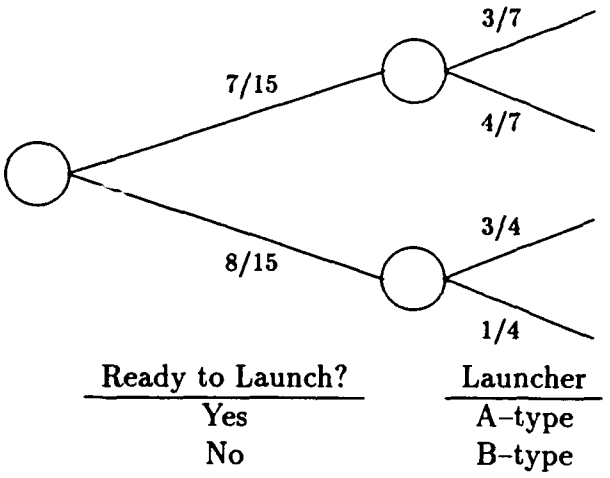
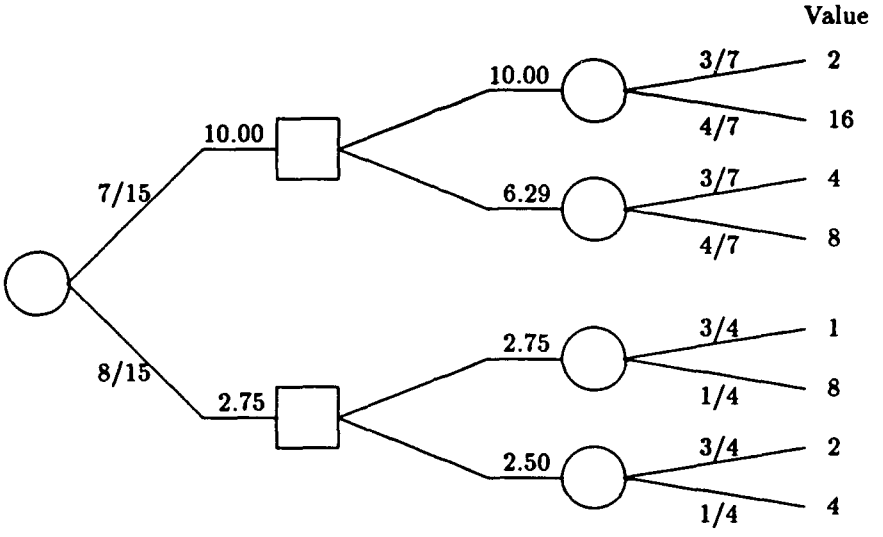


Figure 3.2. Second Decision Tree

Figure 3.2 cannot help with the recommendation on attack type. Further, it does not depict an important part of the information—the attack values. Adding those pieces of information results in the complete tree in Figure 3.3.

From Figure 3.3, the designer can make the best recommendation about attack type. The useful information for this decision comes from the lower half of the tree because the launcher does not indicate ready for launch. In the lower half of the tree, it is clear that the expected attack value using a missile (2.75) exceeds the



Ready?	Attack	Launcher
Yes	Missile	A-type
No	Strafe	B-type

NOTATION: In columns 2 and 3, the numbers immediately preceding the nodes are expected values computed from the numbers in the "Value" column. The square nodes in column 2 are "decision nodes"; the circular nodes elsewhere are "condition nodes".

Figure 3.3. Final Decision Tree

expected attack value using a strafing pass (2.50). Hence, the pilot should attack with a missile.

3.3.2 Discussion of the Solution to This Problem The designer of the moving map display has a dilemma. The most likely identity of the target is A–type, and the recommended ordnance is associated with the B–type launcher. The designer must decide: should the map display show the highest probability identity and also the "inconsistent" attack recommendation? This seeming contradiction is characteristic of the problem, not the solution technique. The ordnance recommendation takes into account more information than just probability of a given identity. It uses the target utility to further account for all the concerns captured in the utility values. In this problem, the high utility associated with attacking a B–type launcher drove the solution. The utility values may have captured other important factors, like risk to the fighter, target value compared to other targets, armament available, etc. If the utility values reflect the utility values of the decision-maker, the missile pass is appropriate.

This lack of correspondence between probable identity and recommended action is important to decision-makers other than war-fighters. If weapon system designers believed that strafing ground targets would be a seldom-used tactic, they might better balance the effort and expense expended on developing the system; they might de-emphasize the strafing capability and better develop the missile-delivery capability. If logisticians believed fighters would make more missile passes than strafing passes, their decisions about purchasing and shipping of supplies might better support the wartime needs. If fighting commanders followed this advice, they would have the best probability of getting results in line with the established values and their weapons load crews could load the weapons the pilots need most.

Having described the solution to this problem, and recognizing that a choice on display symbology lies properly with the system designer, this thesis must leave the important issue of reconciling this problem to another work.

3.4 Influence Diagram Solution

3.4.1 Basics of Influence Diagrams Influence diagrams are another technique for solving the same problem. Like decision trees, influence diagrams are directed graphs. Influence diagrams have three types of nodes: "chance nodes", "decision nodes", and "value nodes" (Howard 89). Chance nodes represent events over which the decision maker has no control. Decision nodes represent decision events. Value nodes represent the output of the tree. Arrows between nodes depict "relevance". Thus, an arrow between two chance nodes documents that the first event is relevant to the distribution of the second. Said another way, the distribution of the second chance node is dependent on the output of the first chance node. Similarly, arrows into a decision node represent information expected to be known to the decision maker at decision time.

Analysts usually depict influence diagrams only with graphics, as shown here. However, the information on distributions (shown here in separate tables) is also integral to the problem solving process. For example, Figure 3.4 is part of the influence diagram for the A-type/B-type problem in this chapter. It depicts that the identity of the launcher of interest is "relevant" to whether the launcher is "ready". Tables 3.7 and 3.8 are the associated distributions. Table 3.6 explains the notation for the distributions.

Chance nodes, depicted in this thesis with a circle, always have a distribution associated with them. The associated distribution is dependent on variables associated with each predecessor node in the diagram. Thus, in Figure 3.4, the "ID" node is not dependent on any external variable (Table 3.7) and the "READY" node is dependent on identity (Table 3.8).

Associated distributions: Tables 3.7 and 3.8

Figure 3.4. Part of Influence Diagram for the A–Type/B–Type Problem

LEGEND	
A–type	subject launcher is A–type
B–type	subject launcher is B–type
missile	attack ordnance: air–to–ground missile
strafe	attack ordnance: strafing pass
"Ready"	subject launcher indicates it is ready for launch
"Not Ready"	subject launcher indicates it is NOT ready for launch
row	possible values of the output variable
columns	values associated with possible values of the input variables

Table 3.6. Legend for Distributions Shown in Tables 3.7 through 3.14

Identity	Probability of Identity
A–type	3/5
B–type	2/5

Table 3.7. Distribution Associated with "ID" Node in Figure 3.4, a "Prior" Distribution

	Identity	
Readiness	A–type	B–type
"Ready"	1/3	2/3
"Not Ready"	2/3	1/3

Table 3.8. Distributions Associated with "READY" Node in Figure 3.4, "Conditional" Probability Information

Part of the theory of influence diagrams includes a list of four transformations analysts may use on the diagrams. Analysts call those transformations

- arc reversal using Bayes' Theorem,
- summing a variable out of the joint [distribution],
- removing a chance node by expectation, and
- removing a decision node by maximization.
  (Tatman and Shachter 90:367)

A pivotal idea of influence diagrams is that analysts may apply the transformations systematically to alter the diagram into a form most useful for answering the question at hand. This section demonstrates solving the influence diagram for the A–type/B–type problem.

3.4.2 Arc Reversal Using Bayes' Theorem Tatman and Shachter identify "arc reversal using Bayes' theorem" as one of their four transformations used on influence diagrams (Tatman and Shachter 90:367). For two chance nodes with one relevance arrow between them (as in Figure 3.4), this transformation allows the user to compute a new probability distribution and reverse the relevance arrow (as in Figure 3.5). Bayes' Theorem shows how to compute, for example, $Pr[I|R]$ from $Pr[R|I]$ and $Pr[I]$. Figure 3.5 represents the same information as Figure 3.4 in a different presentation.

Associated distributions: Tables 3.9 and 3.10

Figure 3.5. Influence Diagram With Relevance Arrow Reversed

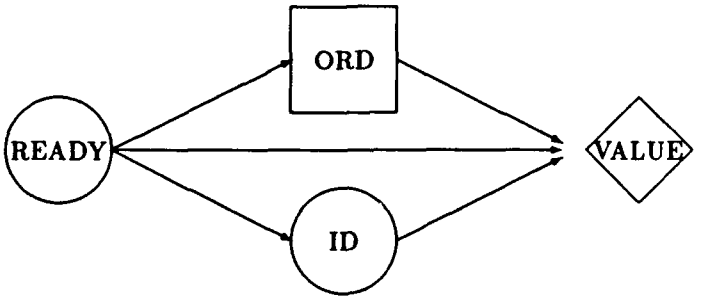
Readiness	Probability of Readiness
"Ready"	7/15
"Not Ready"	8/15

Table 3.9. Distribution Associated with "READY" Node in Figures 3.5 and 3.6, a "Preposterior" Distribution

	Readiness	
Identity	"Ready"	"Not Ready"
A–type	3/7	3/4
B–type	4/7	1/4

Table 3.10. Distributions Associated with "ID" Node in Figures 3.5 and 3.6, "Posterior" Probability Information

If the decision analyst builds the full influence diagram for the A–type/B–type problem around Figure 3.5, the result is Figure 3.6. This completed diagram includes all three of the node types.



Associated distributions: Tables 3.9, 3.10, and 3.11

Figure 3.6. Full Influence Diagram for the A–Type/B–Type Problem

Decision nodes, depicted in this thesis with a square, have no associated distribution. Rather, decision nodes imply the use of a decision rule, usually utility maximization, to solve the node out of the influence diagram.

Readiness	Identity and Ordnance			
	A-type		B-type	
	missile	strafe	missile	strafe
"Ready"	2	4	16	8
"Not Ready"	1	2	8	4

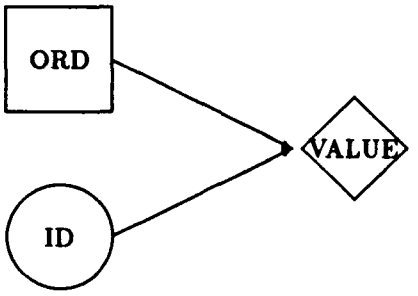
Table 3.11. Utility Lookup Table For "Value" Node in Figure 3.6

Value nodes, depicted in this thesis with a diamond, have a utility function associated with them. Like the distribution on chance nodes, this function has as many input variables as there are predecessor nodes in the diagram. Thus, in this example, the distribution associated with the value node has three input variables—"READY", "ID", and "ORDNANCE"—as in Table 3.11.

3.4.3 Starting the Solution Process While Figure 3.6 correctly shows the influence diagram upon which to base analysis, it does not reflect all information given in the problem. In particular, the "READY" node no longer should have a probability distribution associated with it—the problem states that the launcher of interest does not indicate ready to launch.

This modification propagates quickly through the diagram. Only four values in Table 3.11 are relevant to this solution—those holding information for launchers not ready to launch. Similarly, only part of the existing distribution for the "ID" node (Table 3.10) is relevant—the part holding information on launchers not ready to launch.

By changing those two distributions, the "READY" node becomes unneeded. Figure 3.7 results. At this point in the analysis, analysts can answer the first question in the A-type/B-type problem: the most likely identity of the launcher is "A-type", with an associated probability of 75 percent (Table 3.12).



Associated distributions: Tables 3.12 and 3.13

Figure 3.7. Influence Diagram for the A–Type/B–Type Problem After Using Known Readiness Information

Identity	Probability of Identity
A–type	3/4
B–type	1/4

Table 3.12. Distribution Associated with "ID" Node in Figure 3.7

	Identity and Ordnance			
	A–type		B–type	
	missile	strafe	missile	strafe
Utility Value	1	2	8	4

Table 3.13. Utility Lookup Table For "Value" Node in Figure 3.7

3.4.4 Removing a Chance Node by Expectation The "ID" node in Figure 3.7 is now removable using expectation. By multiplying the probabilities of each launcher type (from Table 3.12) by each corresponding utility value in the utility table (Table 3.13), the analyst can create an expected utility table and delete the "ID" node. Figure 3.8 results.

Associated distribution: Table 3.14

Figure 3.8. Influence Diagram for the A–Type/B–Type Problem Using Expectation to Delete "ID" Node

	Identity and Ordnance			
	A–type		B–type	
	missile	strafe	missile	strafe
Expected Utility	0.75	1.50	2.00	1.00

or, reordering

	Ordnance and Identity			
	missile		strafe	
	A–type	B–type	A–type	B–type
Expected Utility	0.75	2.00	1.50	1.00
Total	2.75		2.50	

Table 3.14. Expected Value Lookup Table For "Value" Node in Figure 3.8

3.4.5 Removing a Decision Node by Maximization The decision node ("ORD") in Figure 3.8 is a simple decision between two types of ordnance. The decision maker decides this issue based on maximum expected value. Simple addition of the expected value associated with each ordnance shows that using a missile gives 2.75 units of value, compared with the 2.50 units of value for using a strafing pass

(Table 3.14). The decision maker maximizes utility by choosing the missile, thus completing the second part of the A–type/B–type problem. As a result of making the decision, the "ORD" node goes away, leaving only the "VALUE" node (Figure 3.9), which shows a fully solved tree.

Figure 3.9. Fully Solved Influence Diagram

Like the analyst using decision trees, the analyst using influence diagrams decided to recommend a missile attack. Both analysts made that recommendation even though the odds were three to one that the launcher was the "A–type" launcher, for which a strafing pass would be twice as desirable. The reader can confirm the mathematical equivalence of the two methods by noting that the numbers used to make the recommendations are identical.

3.5 Probability Ratio Nets

3.5.1 Basics of Probability Ratio Nets Probability ratio nets (PRN's), due to Morlan (Morlan 91), are a newer method of solving this problem. Like influence diagrams, his technique involves building nets using the information available with the intent of transforming them into more useful form. PRN's are directed graphs with no cycles. Each node represents an event. The arc between two nodes depicts the relative probability of the two events. Thus, in Figure 3.10, event A–type is half again as likely as event B–type.

Nodes can contain PRN's, creating a hierarchical structure. In the A–type/B–type problem, an analyst would add hierarchy to Figure 3.10 by embedding PRN's showing the probability of each type launcher being ready for launch. Figure 3.11 presents the result.

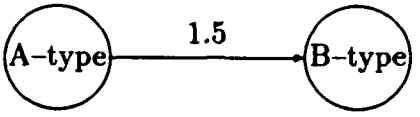


Figure 3.10. Probability Ratio Net: Event "A-Type" Is Half Again as Likely as Event "B-Type"

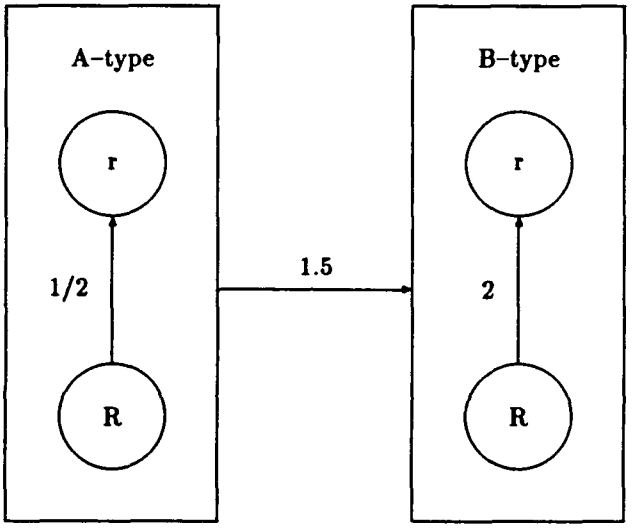


Figure 3.11. Initial Probability Ratio Net for the A-Type/B-Type Problem

LEGEND	
A-type	subject launcher is A-type
B-type	subject launcher is B-type
R	subject launcher indicates ready for launch
r	subject launcher indicates not ready for launch
AR	subject launcher is A-type and ready
Ar	subject launcher is A-type and not ready
BR	subject launcher is B-type and ready
Br	subject launcher is B-type and not ready

Table 3.15. Legend for Node Labels in Figures 3.11 through 3.22

Figure 3.11 has six nodes: one "A-type" node, one "B-type" node, two "R" nodes, and two "r" nodes. The nodes marked "A-type" and "B-type" contain other nodes—each contains one "R" node and one "r" node. Within the "A-type" node, the readiness probabilities apply only to A-type launchers. Since the analyst needs to know the probabilities of launcher type based on whether the subject launcher is ready for launch, the analyst needs to transform Figure 3.11 into Figure 3.12. At the position of the question mark in Figure 3.12, the analyst will read the probability ratio between launcher types given that the subject launcher is not ready for launch. This section shows the transformation.

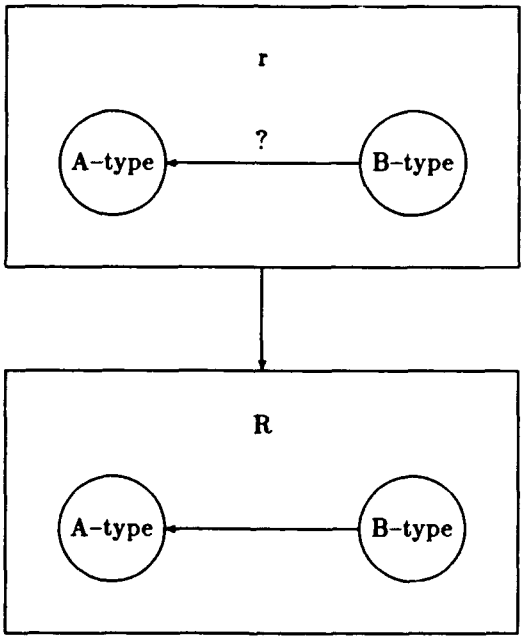


Figure 3.12. Desired Probability Ratio Net for the A-Type/B-Type Problem

3.5.2 Reversing an Arc Sometimes, an analyst needs to reverse the direction of an arc. Referring to the information in the example about readiness of A-type launchers, it seems intuitive that if A-type launchers are half as likely to be ready, they must be twice as likely to be not ready. Morlan proves the assertion (Morlan 91).

By making this change, Figure 3.11 becomes Figure 3.13. The two nets are different representations of identical information.

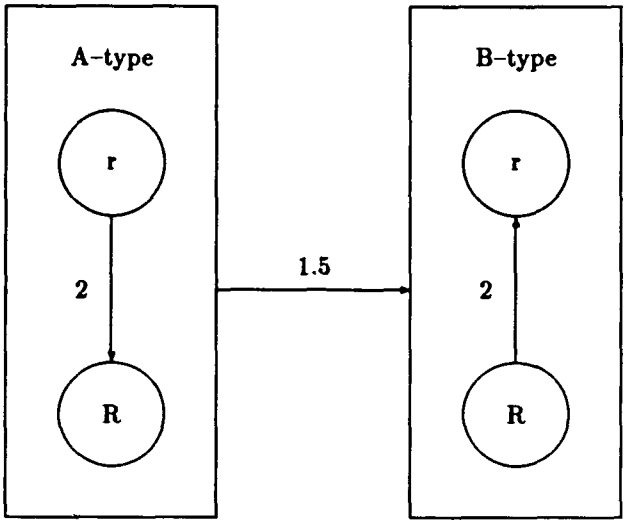


Figure 3.13. Probability Ratio Net for the A–Type/B–Type Problem, With Arc Reversed Inside the "A–type" Node

3.5.3 Forming the Joint Sometimes, an analyst needs to eliminate some of the hierarchy in a net. Morlan calls his tool for this purpose "forming the joint" (Morlan 91). Figure 3.14 is based on his work.

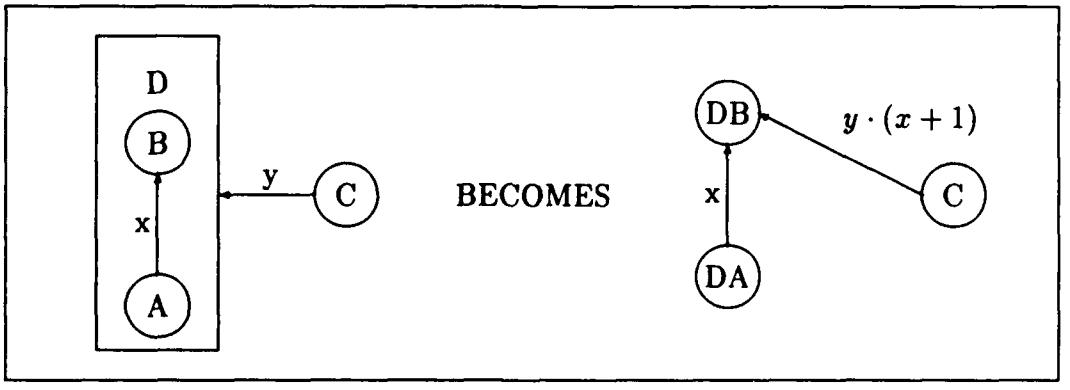


Figure 3.14. Probability Ratio Graph Forming the Joint

The right side of Figure 3.13 is ready for forming the joint. After the operation, Figure 3.15 results. All the same information is in the new graph with a different presentation. In particular, this graph shows there is 9/2 as much chance of the subject launcher being "A-type" (without regard to readiness) as the subject launcher being both "B-type" and not ready.

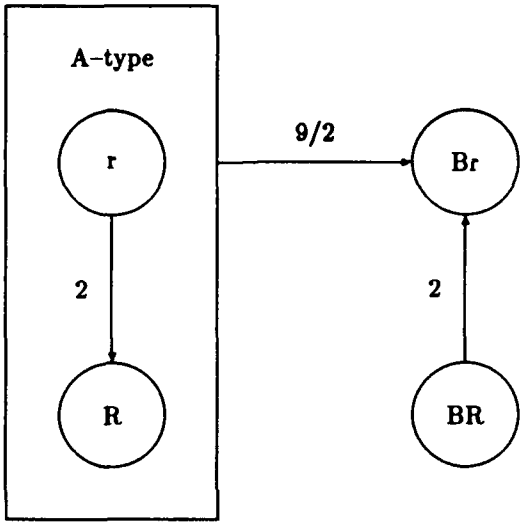


Figure 3.15. Probability Ratio Net for the A-Type/B-Type Problem After Forming the Joint on the Right Side

The analyst also needs to eliminate the remaining hierarchy. By reversing the arc connecting the left side to the right (indicating there is 2/9 as much chance of the subject launcher being both "B-type" and not ready as being "A-type"), the left side of the net becomes ready for forming the joint. (The revised net is not depicted). After forming the joint, the net takes the form shown in Figure 3.16.

3.5.4 Triangulation Sometimes, an analyst needs to have arcs in the net which are other than those present. Morlan calls his tool for this purpose "triangulation" (Morlan 91). Figure 3.17 is based on his work.

In this example, the analyst needs to create a net with only a single arc having vertical travel. Noting that the three nodes in the lower right corner of Figure 3.16

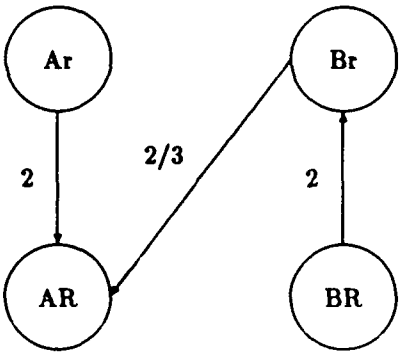


Figure 3.16. Probability Ratio Net for the A-Type/B-Type Problem After Forming the Joint on the Left Side

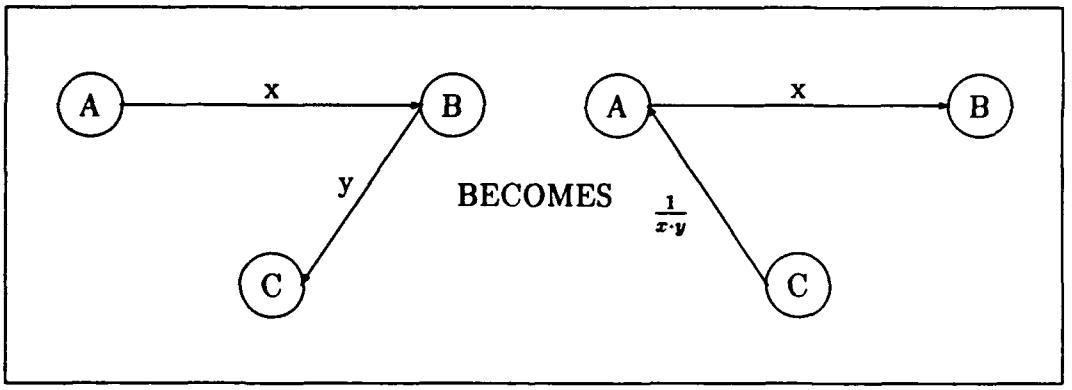


Figure 3.17. Probability Arc Triangulation

are ready for triangulation, the analyst can remove the arc from "BR" to "Br" and add an arc from "AR" to "BR." Figure 3.18 is the result.

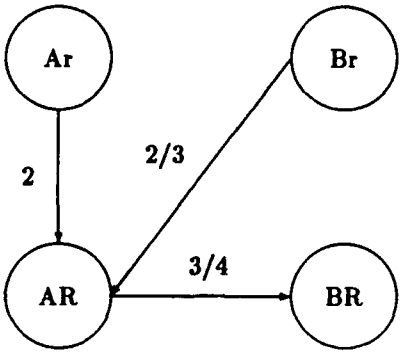


Figure 3.18. Probability Ratio Net for the A–Type/B–Type Problem After Triangulation of the Lower Right Nodes

To eliminate another arc with vertical travel, the analyst may reverse the arc between "Ar" and "AR", thus preparing the three nodes in the upper left corner of Figure 3.18 for triangulation. (The revised net is not depicted.) After triangulating, Figure 3.19 results.

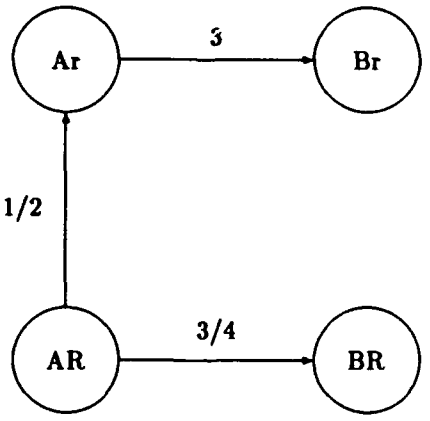


Figure 3.19. Probability Ratio Net for the A–Type/B–Type Problem After Triangulation of the Upper Left Nodes

3.5.5 Conditioning Sometimes, the analyst needs to add hierarchy to a net. Morlan calls his tool for this purpose "conditioning" (Morlan 91). Figure 3.20 is based on his work.

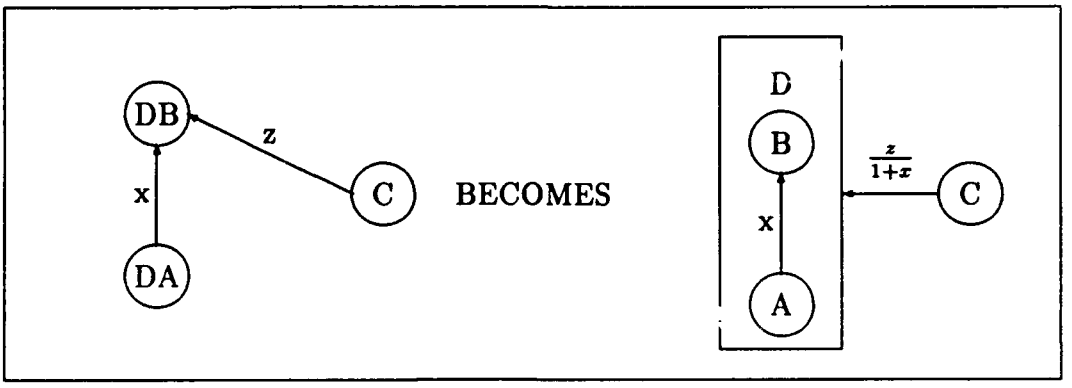


Figure 3.20. Probability Ratio Graph Conditioning

In this example, the analyst wants to add hierarchy to the net by grouping the top two nodes and by grouping the bottom two nodes. By reversing the arc between "Ar" and "Br", the net becomes ready for conditioning of the upper nodes. (The revised net is not depicted.) Figure 3.21 is the result of the conditioning.

By reversing the two un-embedded arcs, the net becomes ready for conditioning of the lower nodes. (The revised net is not depicted.) Figure 3.22 is the result of the conditioning.

3.5.6 Forming a Probability Distribution From a PRN Figure 3.22 gives the analyst the desired information for the first part of the problem: given that the subject launcher is not ready for launch, there is 1/3 the chance the launcher is B–type as A–type. The figure does not give a probability distribution, but computing that distribution is easy. Where $P(A)$ is the probability of an A–type launcher, and $P(B)$ is the probability of a B–type launcher, the top half of Figure 3.22 tells the analyst that after assuming "r", the ratio $\frac{P(B)}{P(A)}$ is $\frac{1}{3}$. Since there are only two types

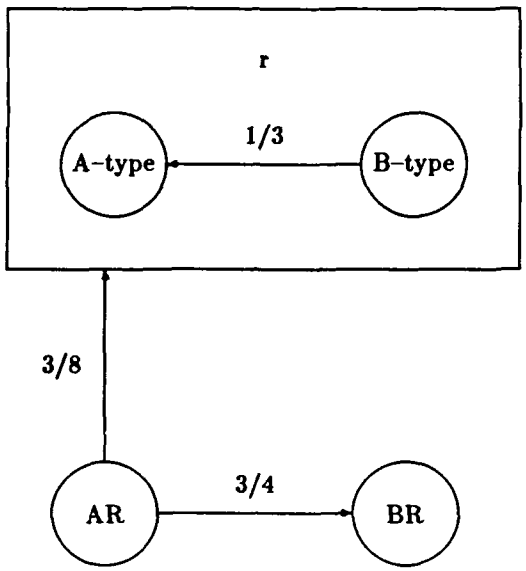


Figure 3.21. Probability Ratio Net for the A–Type/B–Type Problem After Conditioning on the Upper Nodes

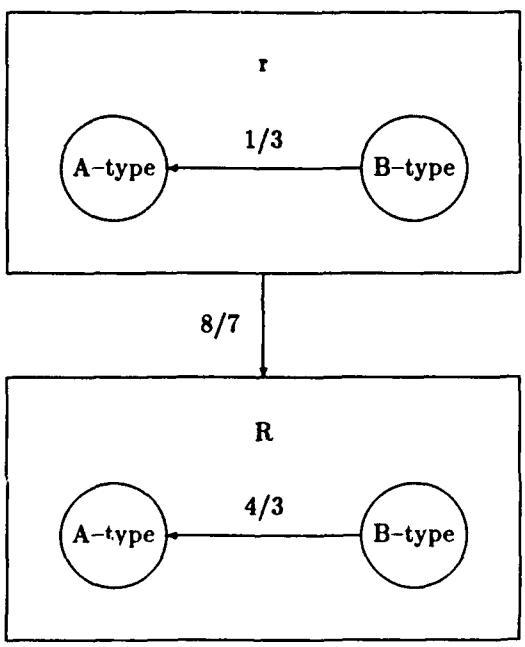


Figure 3.22. Probability Ratio Net for the A–Type/B–Type Problem After Conditioning on the Lower Nodes

of launchers and since the analyst is forming a probability distribution,

$$P(A) + P(B) = 1$$

Then using the above information, the following series of equations demonstrates solving for $P(A)$ and $P(B)$.

$$P(A) + P(A) * \frac{P(B)}{P(A)} = 1$$

$$P(A) * (1 + \frac{1}{3}) = 1$$

$$P(A) = \frac{3}{4}$$

$$P(B) = P(A) * \frac{P(B)}{P(A)} = \frac{3}{4} * \frac{1}{3} = \frac{1}{4}$$

In a check for consistency, the reader can confirm from Figure 3.2 that the two probabilities $P(A)$ and $P(B)$ are 1/4 and 3/4, respectively, a ratio of 1/3. Other values in Figure 3.22 are consistent with Figure 3.2, too.

Similar logic produces the means of translating a PRN with more than two nodes. Given that the analyst converts the structure of the PRN such that all nodes point to some first node A, then the probability of that node P(A) is

$$P(A) = \frac{1}{\sum_{i=1}^{n} \frac{P_i}{P(A)}}$$

where i is an index across all n nodes and P_i is the probability of the ith node. As above,

$$P_i = P(A) * \frac{P_i}{P(A)}$$

3.5.7 Recommending Action Like Figure 3.2, Figure 3.22 does not take utility information into account. Hence, like Figure 3.2, it cannot help in the second

part of the A-type/B-type problem: suggesting an attack mode. Morlan's work does not extend this process. Rather, he recommends using a simple spreadsheet like Table 3.16 to make the decision (Morlan).

Ordnance	Target Type	A Probability	B Utility	Expected Utility (A X B)	Sum For Ordnance
missile	A-type	0.75	1	0.75	
	B-type	0.25	8	2.00	2.75
strafe	A-type	0.75	2	1.50	
	B-type	0.25	4	1.00	2.50

Table 3.16. Expected Utility Spreadsheet

As with the other techniques, this spreadsheet shows that the recommendation of firing a missile has the greater expected utility. This technique made the same recommendation as the other techniques and produced the same numbers, again demonstrating the mathematical equivalence of the three techniques.

3.6 Conclusion

The field of Decision Analysis is probably better off for having many ways to represent knowledge. Decision trees, though valuable, can be difficult to work with. Influence diagrams, though valuable, are most workable when the user has computer support and software written to support them. To start to build information in the literature on probability ratio nets, this work will apply probability ratio nets.

IV. Logical Structure of the Subject Problems

Designers of action recommendation systems for future aircraft will need some structure on the information they will provide the system and will need an understanding of the processing the system must do. This chapter presents such a structure and a description of the processing. The first part of the chapter defines the structure of the information the system will need. The second part of this chapter describes the processing the system must do to achieve useful recommendations.

Members of the research and development community may take interest in relating these ideas to designs of future fighters. Members of the combat operations community may find this information helpful in understanding the value and theory of designs offered to them.

4.1 Information Groups

4.1.1 Actions The system envisioned for this thesis effort chooses the optimal action from among those on a list. Combat operations planners will prepare exhaustive lists of possible actions and provide those lists to the system. Table 4.1 presents a list of actions for the system, though it is not exhaustive. The action recommendation system will not consider actions not placed on the list, no matter how attractive those actions are.

4.1.2 Identities and the Prior Distribution In a similar way that the action recommendation system gets a list of possible actions, the system gets a list of possible identities for the targets it will analyze. Associated with the list of identities must be information on the expected probability of analyzing targets of each identity. This list of identities and probabilities is likely to be a joint responsibility of operations planning personnel and intelligence personnel.

attack immediately with air-to-ground missile; line formation
attack immediately with air-to-ground missile; wedge formation
attack immediately with air-to-ground missile; trail formation
strafe immediately; [each formation]
delay forward motion of the target
turn toward: collect further information
turn away: possible hostile; no attack
probable non-combatant (no course change necessary)

Table 4.1. Example Possible Actions List

If the planners build an identity list and associated distribution for a time in a mission when the fighter's sensors assess targets at a civilian truck stop, Table 4.2 might represent their product. In that table, the probabilities associated with civilian vehicles are higher than the probabilities associated with military vehicles. Probabilities associated with friendly and enemy combat vehicles are roughly equivalent, reflecting uncertainty about control of the area.

Identity	Percent Probability of Identity
T-72 tank	3
T-80 tank	3
enemy multiple rocket launcher	5
BRDM (enemy armored personnel carrier)	3
SA-8 (surface-to-air missile)	2
SA-13 (surface-to-air missile)	2
M-60 tank	5
Bradley (U.S. armored personnel carrier)	2
fuel truck	15
passenger bus	20
heavy truck	40

Table 4.2. Example Identity and Probability Information

In any system in a future air-to-ground fighter, the idea of programming the computers with multiple prior distributions, each applied at appropriate times, is important. Probably, the example of Table 4.2 would be inappropriate to use if the fighter were flying over a known enemy armor base, because the probabilities of enemy combat vehicles should be higher and the probability of civilian vehicles should be lower. The probability of friendly vehicles should probably remain low. A recommendation system which is responsive to the situation is most appropriate. Chapter 5 contains examples explaining this idea more fully.

4.1.3 Sensor Suite Characteristics The action recommendation system must have a description of the sensor suite available. The research and development community should deliver these values to operators with the sensor suites they build.

For purposes of this thesis, all sensors report from defined lists. Table 4.3 represents a sensor suite description. For example, the sensor suite it represents has the ability to discriminate whether the target vehicle has wheels, with two distinct reports possible. Such a feature might be valuable for distinguishing armored personnel carriers from vans. Similarly, the suite can report how many windows are distinguishable on the target vehicle, with up to twenty distinct reports possible. Such a feature might be useful for distinguishing long fuel trucks from buses. The suite can report classes of length-to-width ratios, with up to twenty classes reportable. The suite might include sensors not on the individual fighter. For example, the report of nearby combat support elements might require input of information collected by a satellite sensor. The system will make its recommendations from reports by each of these sensors.

A single sensor might produce multiple reports, which this system could treat as multiple sensors. Singstock (Singstock 91) used a single infrared sensor to develop multiple features of his targets. If a system like Singstock's would report those features to an action recommendation system, the action recommendation system

Sensor Attribute	Number of Distinct Reports
target has wheels	2
target has treads	2
number of windows	20
target length-to-width ratio	20
target size	20
engine mounting position	3
engine type	4
combat support elements nearby	3

Table 4.3. Example Sensor Suite Description

could convert those continuous features into reports of membership in one of several ranges. Thus, the readings would effectively become discrete. Then, the action recommendation system could treat each feature as a sensor.

4.1.4 Identity-Sensor Data The system needs to know the expected response of each identity to the sensor suite. In terminology associated with Bayes' Theorem, the system needs to know a conditional probability distribution for each possible identity. That is, for i possible identities, and for j possible reports from the sensor suite, the system must know the probability of each report given each identity, $P(Report_j|Identity_i)$. The intelligence community is likely to be responsible for these values.

Tables of identity-sensor data can grow large. Consider a problem with only two possible identities ("T1" and "T2") and with a sensor suite composed of two sensors. If one sensor reports either "A" or "B", and the other sensor reports either "C" or "D", the sensor suite has four possible reports: "AC", "AD", "BC", and "BD". Table 4.4 might represent the identity-sensor data for the problem; each column is a probability distribution. These tables can grow in size quickly—the number of entries in a table for a problem with twice as many sensors and twice as many reports on each sensor would have 32 times the number of entries in Table 4.4.

Sensor Suite Report	Identity	
	T1	T2
AC	.2	.5
AD	.1	.1
BC	.2	.1
BD	.5	.3

Table 4.4. Example Identity-Sensor Data: Probability of Each Sensor Report Given Each Identity

4.1.5 Utility Data The system needs to be able to build utility numbers for each combination of action, identity, and sensor suite report. These values are likely to be the responsibility of combat operations planners. In building the values, the planners would be responsible for making the values reflect the fighting policies of the operation commander.

Tables of identity-utility data can grow large. For the identity-sensor data shown in Table 4.4, if there are only two actions to consider ("E" and "F"), the system must be able to build a table like Table 4.5.

Sensor Suite Report and Action	Identity	
	T1	T2
AC, E	3	1
AD, E	4	2
BC, E	6	1
BD, E	8	7
AC, F	7	2
AD, F	5	6
BC, F	1	4
BD, F	5	4

Table 4.5. Example Utility Data: Utility Values

The numbers in Table 4.5 indicate relative value. Any one raw utility number is meaningful only when compared to another utility value. Thus, if all numbers in

Table 4.5 were double the values shown, the table would hold identical information and support identical decisions.

4.1.6 Sensor Report All the information mentioned up to here is the responsibility of some staff agency. All that information would be available to the fighter's action recommendation system before the system needed to make recommendations, potentially before takeoff. The sensor report is distinct; it represents the report of the sensor suite about the scene ahead. For the purposes of this thesis, the sensor report is a string of characters, with each character representing the report of a single sensor. In the sensor suite described for Tables 4.4 and 4.5, examples of reports include "AC" and "BD". This represents information today's pilots must sift through and information the fighter of the future can process in a way to help the tomorrow's pilots.

4.2 Processing the System Information

There are three major tasks any system solving this class of problems must accomplish to provide useful recommendations to the pilot. The first task is translating the inputs from their input form into their internal form for use of the system. The discussion in this section of this first task concerns only the desired output because other details are strongly dependent on implementation details. The second task of the system is to develop an assessment of the possible identities of the target. This task is the same work mentioned in Chapter 3 as the first half of the example problem worked; the output is a probability distribution showing the probability of each possible identity. The third task of the system is to develop an assessment of the best action to take. This task is the same work mentioned in Chapter 3 as the second half of the example problem worked. This section discusses parts of each task.

4.2.1 Internal Form of the Data As the system develops its assessment of identity and its recommendation, only some of the data available to it is relevant. In particular, the system needs two arrays of data. One array records the interactions between the sensor suite report and the possible identities; the other array records the utilities associated with the sensor suite report.

4.2.1.1 The Sensor Suite Report and the Identities Obviously, information applying to all but one of the possible reports of the sensor suite is irrelevant. That is, if the sensor suite is reporting a report X, then for any other report Y, the information about what the target could be if the report was Y is of no interest. The system can concentrate in particular on report X and can group all reports Y into a single class of reports which the system will ignore.

With probability ratio nets (PRN's), a convenient way to represent this information is with two nodes, as in Figure 4.1. The source of this information is the Identity-Sensor Data mentioned above. The information in Figure 4.1 applies to one particular identity; the system needs a series of these PRN's, using one for each possible identity. Hence, the system can build a PRN like Figure 4.2. Information for the ratios between nodes of each identity comes from the Prior Distribution.

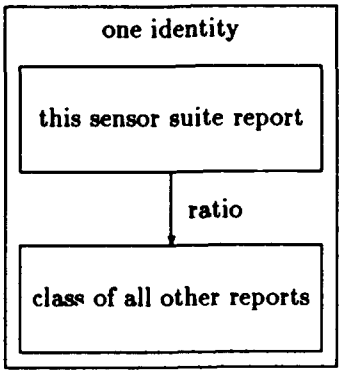


Figure 4.1. Probability Ratio Net Indicating Relative Probability of the Sensor Suite Report to Class of All Other Reports

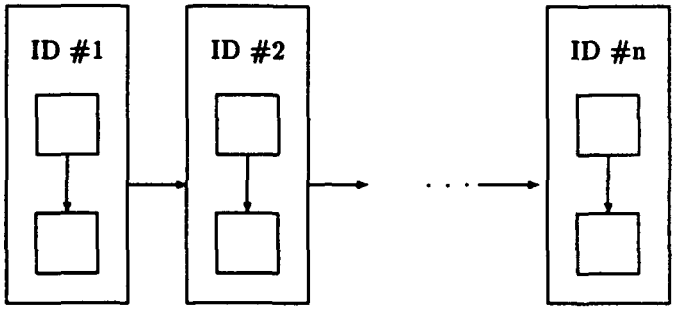


Figure 4.2. Probability Ratio Net Connecting Nodes for All Identities

4.2.1.2 Utilities Associated with the Sensor Suite Report For the utility information, the system needs the information corresponding to the sensor suite report. Information for other possible reports is irrelevant. Thus, for the example mentioned in Table 4.5, most of the information is not needed. If the sensor suite report is, for example, "BD", Table 4.5 reduces to Table 4.6 by the elimination of lines for other sensor reports.

Sensor Suite Report and Action	Identity	
	T1	T2
BD, E	8	7
BD, F	5	4

Table 4.6. Example Utility Data for a Single Report; A Subset of Table 4.5

4.2.2 Processing Tasks

4.2.2.1 Assessing the Identity The job of assessing a target's identity is to produce a probability distribution indicating the probability of each identity. The input to the task is the probability ratio net in Figure 4.2. To generate the required probability distribution, the system converts Figure 4.2 to Figure 4.3. In point of fact, the particular PRN depicted in Figure 4.3 is not necessary. The ratios r the system needs are depicted r_1 through r_{n-1} in Figure 4.3 and any PRN with those

numbers is equally useful, without regard to the structure of the other nodes. The system converts those identified ratios r to a probability distribution as discussed in paragraph 3.5.6.

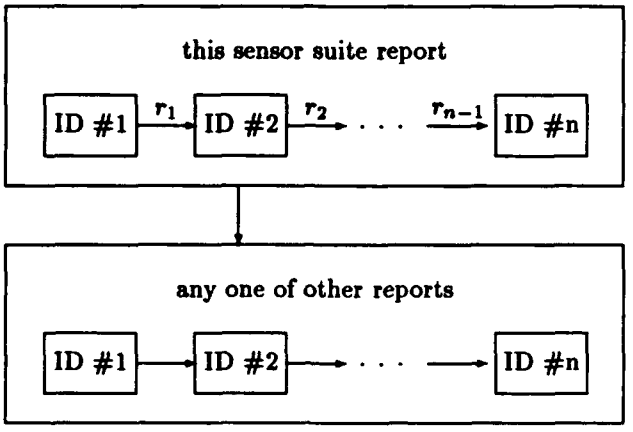


Figure 4.3. Probability Ratio Net Holding Needed Ratios

4.2.2.2 Recommending Action Once the system has solved for the probability distribution on the identities and developed the utility table in Table 4.6, the job of recommending actions becomes one of computing the expected utility of each action and reporting the action with the maximum utility.

Computing the expected utility is easy. In Table 4.6, each row contains utilities for one action and all possible identities. After multiplying each of those utility numbers by the corresponding probability of the identity, the sum of the products is the expected utility of the action. For example, if the probability distribution associated with Table 4.6 is the distribution presented in Table 4.7, Table 4.8 represents solving the problem. Utility theory requires that the decision-maker choose the action with the highest expected utility—in this case, "E" (with 7.4) rather than "F" (with 4.4).

Identity	Probability
T1	.4
T2	.6

Table 4.7. Example Identity Distribution for Table 4.6

Action	ID	A Utility (Table 4.6)	B $P(ID)$ (Table 4.7)	$A * B$ Expected Utility	(sum) Expected Utility
E	T1	8	.4	3.2	
	T2	7	.6	4.2	7.4
F	T1	5	.4	2.0	
	T2	4	.6	2.4	4.4

Table 4.8. Computation of Expected Utility for Each Action

V. *Solution and Results*

Earlier chapters described ideas; this chapter briefly describes an implementation of those ideas and presents examples of solutions. The author designed and completed a computer program to show suitability of probability ratio nets (PRN's) to the subject class of problems. The first section of this chapter introduces the method of using the program code. The second section of the chapter describes the verification and informal validation of the program, and explains why formal validation was not appropriate. The third section of the chapter poses two example problems and presents the results achieved by the program, to include several examples to show the value of utility theory for this class of problem.

5.1 The Thesis Code

5.1.1 Using the Thesis Code To use the thesis code, users create the needed data files and use a short Turbo Pascal program to process them. The author used version 6.0 of the Turbo Pascal compiler. The code has worked with Turbo Pascal version 5.5 without difficulty; the author has not tested compatibility with other versions of the compiler.

Table 5.1 is a full user program using the thesis code. The user program gains access to the thesis code via the "USES ProjModU" statement. The thesis code provides a single procedure for use; its name is "ProjMod". Procedure "ProjMod" has six input parameters and two output parameters. Each of the six inputs is a file name. Table 5.1 shows extensions in the correct places; the extensions are allowed but the extensions are not required.

The variable "MostLikelyID" returns the number in the list of possible identities of the identity with the highest probability of representing the target. The lowest number returned is 1; the highest number returned is the number of possible identities.

```
{$M 65520,0,655360} { - allows use of maximum memory for both
                      stack and heap }

PROGRAM ThesisCaller;

USES
  ProjModU;

VAR
  MostLikelyID,
  RecommendAction: INTEGER;

BEGIN { - of PROGRAM Caller }

  ProjMod ( 'TEST.PID' , 'TEST.PAC' , 'TEST.ESP' ,
    'TEST.ROW' , 'TEST.PRR' , 'TEST.USF' ,
    MostLikelyID , RecommendAction );
  WRITELN ( 'MostLikelyID = ' , MostLikelyID );
  WRITELN ( 'RecommendAction = ' , RecommendAction );

END.
```

Table 5.1. Sufficient Source Code to Use the Thesis Code

The variable "RecommendAction" returns the number in the list of actions of the action with the highest expected utility. The lowest number returned is 1; the highest number returned is the number of actions. Appendix A is a detailed user's guide for the program. Appendix B presents the development documents for the code. Appendix C presents the code.

5.2 Verification and Validation

5.2.1 Verification The author verified the code operates as designed in two ways. First, the author checked all routines for proper operation with detailed inspection of all data structures during the development phase of the project. Each routine appears to function as designed and the routines appear to work together to create the planned output. Second, the author built a spreadsheet to develop the same assessments as the program. This spreadsheet is useful for problems with up to five possible identities, with up to five sensors each reporting one of two reports, and with up to five actions. The two problems presented in this chapter stay within those limits, so the author solved these two problems using identical input for the thesis code and for the spreadsheet. The two methods produced the same identity assessments and the same recommended actions. More important, the two methods computed the same probability distributions on the identities for every variant of the two problems tested. Further, the two methods computed the same expected utility numbers for all actions and for all variants tested. The code and the spreadsheet appear to achieve identical results and those results appear to be the desired results.

More thorough verification is possible, but deemed unnecessary. For example, the code could be checked for indexing problems for the largest problems. Similarly, the code could be checked for adherence to each limit written in the specification documents. For example, the specification documents impose a limit of 65535 separate possible sensor reports, and the code checks the "expected sensor performance" file for adherence to the limit, but the code has not been formally

tested. This type of testing was deferred for this project because this code is not production code. This code is useful primarily for proof of concept.

5.2.2 Validation The variations of Example 2, the "T/APC/SAM Problem" later in this chapter, show an informal validity. They show that the output makes sense (as measured by a standard of face validity) and that changes in inputs create expectable changes in output. The code is not formally validated—no one has confirmed the code makes the "right" recommendation given real data. The purpose of this thesis is to prove the concept of using utility theory in the environment of the air-to-ground fighter of the future. No real data exists. Much work with sensor design and cockpit design remains before validation of a project like this one will be possible.

5.3 Example Problems and Results

This section presents two example problems and discusses them. The first example problem is the simple A–type/B–type problem presented in paragraph 3.1. This section presents the code's solution to that problem as a means of confirming operation of the code on a problem known to the reader. The second problem is more complex. That section includes solutions to several variations of the more complex problem, each chosen to show the value of the utility theory approach to the design of future fighter aircraft.

5.3.1 Example 1: The A–Type/B–Type Problem From Paragraph 3.1 The first part of this section is a problem statement. Following the problem statement are the files the author constructed to solve the A–type/B–type problem with the thesis code. The information in this section on constructing those files is very brief; full documentation is in Appendix A.

5.3.1.1 Problem Statement (Repeated From Chapter 3) At a time of interest, an enemy has just launched a surface-to-surface missile. There are two variants of the launcher: the "A-type" and the "B-type". Intelligence reports suggest that near the launch location the enemy employs half again as many of the A-type missile launchers compared to the B-type launcher.

The launcher types differ in reaction time and defensive armament. The B-type launcher prepares itself for missile launch faster, so at the time of interest, it is twice as likely to be ready for launch as to be not ready. At the time of interest, the situation for the A-type launcher is reversed—it is twice as likely to be not ready for launch as to be ready. In a second difference between the two types of launcher, the B-type launcher has added defensive equipment which increases the risk to a fighter making a strafing pass.

Both launcher types emit an identifiable radio signal when the launcher is ready for launch.

The air order of battle places values on attack options. Attacking A-type launchers with a strafing pass is twice as desirable as using a missile. Attacking B-type launchers with a missile is twice as desirable as a strafing pass. A B-type launcher is four times as desirable to destroy as an A-type. Either type launcher is twice as desirable to destroy if it is ready for launch.

An air-to-ground fighter of the future is patrolling this area of the battlefield and seeks to destroy the launcher which has just fired. The fighter's sensor suite has detected a launcher of unknown type and determined that the launcher is not emitting the "ready for launch" signal.

Question Number 1: What Identity to Display? Assuming the fighter has a moving map display which must display a symbol for either "A-type" or "B-type", a problem facing the designer of the map display is to decide which symbol to use.

Question Number 2: What Action to Recommend? A further problem is to recommend which armament to use.

5.3.1.2 The Identities File

```
a-type
b-type
```

This above box presents the identities file. The file lists the file names associated with the two identities.

5.3.1.3 The Actions File

```
strafe
missile
```

This above box presents the actions file. In this simple example, the only decision is what ordnance to use in the attack.

5.3.1.4 The Expected Sensor Performance File This file specifies the performance of the sensor suite. In this simple case, the file contains a single digit, "2". The file specifies a single sensor with two possible reports.

5.3.1.5 The Report of the World File This file specifies the sensor suite report. In this simple case, the file contains a single digit, "1". The file indicates the sensor suite returned the first of the two possible reports.

5.3.1.6 The Priors File

```
3 2
```

The above box presents the file of relative prior probabilities. The file indicates that the probability of the first-listed identity ("A-type") is half-again the probability of the second-listed identity ("B-type").

5.3.1.7 The Utility Scaling Factors File

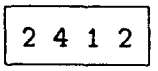
The above box presents the file of utility scaling factors. The file indicates the first-listed identity ("A-type") has one-fourth the utility of the second-listed identity ("B-type").

5.3.1.8 The Identity-Sensor Data Files

The above box presents the file of identity-sensor data for the A-type launcher. It indicates the sensor is twice as likely to return the first possible reading as the second reading if the target is of A-type.

The above box presents the file of identity-sensor data for the B-type launcher. It indicates the opposite of the A-type; the first sensor is half as likely to return the first possible reading if the target is of B-type.

5.3.1.9 The Identity-Utility Data Files



The above box presents the identity-utility data for the A-type launcher. It indicates a utility of "2" for the combination of the first action and the first report: the first report corresponds to strafing an A-type target which is reporting ready to fire. The file indicates a utility of "4" for the combination of the first action and the second report (strafing an A-type target which is reporting ready to fire). It indicates a utility of "1" and "2" for the combinations of the second action and the two reports (corresponding to using an air-to-ground missile).

```
1 2 2 4
```

The above box presents the identity-utility data for the B–type launcher. The B–type file presents utility values in the same order and has different utility values.

5.3.1.10 Program Output

```
MostLikelyID = 1
RecommendAction = 2
```

The above box presents the output of the program in Table 5.1 when that program processes the above files. The first line of output indicates the target is most likely to have the first-listed identity in the identities file ("A–type"). The second line of output indicates the action with the highest expected utility is the action with the second group of values in the identity-utility data files, ("missile"). These results are identical to the results found in Chapter 3.

5.3.2 Example 2: A Problem With Tanks, Armored Personnel Carriers, and a Surface-to-Air Missile (T/APC/SAM) This section poses a more complex example using the names of existing weapon systems. After solving the posed problem, this section discusses related solutions. The related solutions help show the value of the utility theory approach to this problem. All data in the problem is notional; the author developed the data to help in explaining utility theory. As it turns out, the characteristics of the sensor suite show the possibility of some undesirable characteristics for sensor suites. The author, after having recognized the problems, decided to use the chosen values anyway because the discussion helps point out some additional value of utility theory to decision-makers. Any realism in the example is due to (Gaebler) and (Stieven).

5.3.2.1 Problem Statement A fighter of the future is flying near the forward edge of the battle area (FEBA) equipped with a sensor suite of five sensors.

Armament for the fighter includes a gun, an air-to-surface missile, and bomblets. Each armament is capable of neutralizing tanks. The sensors report, respectively, that the target

- has a gun ("G") or not ("g"),
- has large missiles (surface-to-surface missiles) mounted on the vehicle ("M") or not ("m"),
- has a friendly transponder ("F") or not ("f"),
- has a turret ("T") or not ("t"),
- is a big vehicle ("B") or not ("b").

From those sensors, the aircraft must assess whether the target is

- an M-60—friendly tank,
- a Bradley—friendly combat fighting vehicle,
- a T-80—adversary tank,
- a BMP—adversary armored personnel carrier,
- an SA-8—adversary surface-to-air missile.

The aircraft must recommend one of five actions to the pilot. They are

- Shoot an air-to-ground missile at the target.
- Strafe the target.
- Drop bomblets on the target.
- Ignore the target—the target is not a threat.
- Turn away from the target—the target is potentially hostile.

Table 5.2 presents the assessed sensor performance with each type of target. It shows in the upper left corner, for example, that the first sensor is twice as likely to report that an M-60 has a gun ("G") as not ("g"). In the same column, the table shows the first sensor is equally likely to report the Bradley has a gun as not. Thus, when aimed at a Bradley, that sensor does not contribute any useful information.

Target ID	$\frac{P(G)}{P(g)}$	$\frac{P(M)}{P(m)}$	$\frac{P(F)}{P(f)}$	$\frac{P(T)}{P(t)}$	$\frac{P(B)}{P(b)}$
M-60	2	1/3	3	5	5
Bradley	1	1/3	2	1/5	1
T-80	2	1/3	1/3	5	5
BMP	1	1/3	1/2	1/5	1
SA-8	1/4	4	1/5	1/8	1/2

Table 5.2. Identity-Sensor Data for the T/APC/SAM Problem

The Commander's intelligence officer reports that in the area of interest, the distribution of the vehicles of the five types will be in proportion to the numbers in Table 5.3. Those numbers show, for example, that among tanks in the area, the T-80's will outnumber the M-60's by a ratio of 50 to 40.

M-60	Bradley	T-80	BMP	SA-8
40	20	50	20	3

Table 5.3. Assessed Relative Probabilities (Prior Information) for the T/APC/SAM Problem

The Commander has selected base utility values for all combinations of identities and actions. Table 5.4 presents the results. It shows, for example, that the Commander prefers attacking the T-80 with a strafing pass (utility value of 10) to attacking the T-80 with a missile (utility value of 6). The Commander's motivation for constructing the table in this way could be, for example, relative effectiveness of the attacks or relative availability of munitions. The Commander could use any

other criteria or combination of criteria to generate this table. The goal of the table is to accurately reflect the values of the Commander.

	missile	strafe	bomblets	overfly	avoid
M-60	0	0	0	10	0
Bradley	0	0	0	10	0
T-80	6	10	8	4	2
BMP	4	10	9	1	1
SA-8	10	4	8	1	8

Table 5.4. Base Utility For Each Identity-Action Combination for the T/APC/SAM Problem

The Commander has further directed that sensor readings be an input to the utility values used in the system. Table 5.5 presents the Commander's orders. It shows, for example, that the Commander assigns one fifth as much utility to a T-80 if the third sensor reports the target has a friendly transponder ("F") as not. Thus, a report of "gmFtb" for the T-80 must have one fifth the utility of a report of "gmftb". This policy encourages the system to recommend appropriate actions against targets most similar to target characteristics. For the T-80, this policy discourages the system from recommending attacks on the M-60, too. This is because for this sensor suite, the only difference in expected reports for the two tanks is the friendly transponder (Table 5.2). Table 5.6 presents the relative utilities for the T-80, all of which result from information in Tables 5.4 and 5.5. Similar tables exist for the other identities, but this work does not include them.

The Commander has ordered that the action recommendation system use identity weights as presented in Table 5.7. This table indicates the Commander wants

- values for BMP's and Bradley's to have little impact in decisions recommended to pilots (low utility scaling factors),

	"G"	"M"	"F"	"T"	"B"
M-60	2		5	2	2
Bradley			5		
T-80	2	1/2	1/5		
BMP			1/5		
SA-8	1/2	2	1/5	1/2	1/2

Locations with no entry indicate an entry of 1—the combination gets no adjustment.

Table 5.5. Utility Adjustments For Each Identity-Action Combination for the T/APC/SAM Problem

- values for T-80's and M-60's to have much greater impact on decisions recommended to pilots and to have the system equally weight the two tanks (utility scaling factors of "5" for each),
- the SA-8 to have the greatest impact on recommendations offered to pilots (the highest utility scaling factor; twice the next highest factor).

For some particular target, the sensor suite reports, "gMftb". Thus, the report indicates the target

- has no gun ("g"),
- has large missiles mounted on the vehicle ("M"),
- has no friendly transponder ("f"),
- has no turret ("t"),
- is a small vehicle ("b").

Problem Number 1 Assuming the fighter has a moving map display which must display a symbol for the identity of the target, a problem facing the designer of the map display is to decide which symbol to use.

Report	missile	strafe	bomblets	overfly	avoid
gmftb	6	10	8	4	2
gmftB	12	20	16	8	4
gmfTb	12	20	16	8	4
gmfTB	24	40	32	16	8
gmFtb	30	50	40	20	10
gmFtB	60	100	80	40	20
gmFTb	60	100	80	40	20
gmFTB	120	200	160	80	40
gMftb	6	10	8	4	2
gMftB	12	20	16	8	4
gMfTb	12	20	16	8	4
gMfTB	24	40	32	16	8
gMFtb	30	50	40	20	10
gMFtB	60	100	80	40	20
gMFTb	60	100	80	40	20
gMFTB	120	200	160	80	40
Gmftb	12	20	16	8	4
GmftB	24	40	32	16	8
GmfTb	24	40	32	16	8
GmfTB	48	80	64	32	16
GmFtb	60	100	80	40	20
GmFtB	120	200	160	80	40
GmFTb	120	200	160	80	40
GmFTB	240	400	320	160	80
GMftb	12	20	16	8	4
GMftB	24	40	32	16	8
GMfTb	24	40	32	16	8
GMfTB	48	80	64	32	16
GMFtb	60	100	80	40	20
GMFtB	120	200	160	80	40
GMFTb	120	200	160	80	40
GMFTB	240	400	320	160	80

Table 5.6. Relative Utilities For Each Report-Action Combination for the T-80 Identity in the T/APC/SAM Problem

M-60	Bradley	T-80	BMP	SA-8
5	2	5	1	10

Table 5.7. Utility Scaling Factors for the T/APC/SAM Problem

Problem Number 2 A further problem is to recommend to the pilot which action to take.

5.3.2.2 Setting Up Files to Solve This Problem with the Project Code

```
M-60   BRADLEY   T-80   BMP   SA-8
```

The above box presents the possible identities file for this problem.

```
shoot an air-to-surface missile at the target
strafe the target
drop bomblets on the target
overfly the target; no threat
avoid the target; potential threat
```

The above box presents the possible actions file.

```
2 2 2 2 2 % five sensors; two reports each
```

The above box presents the expected sensor performance file.

```
1 % no gun
2 % has large missiles
1 % no friendly transponder
1 % no turret
1 % large
```

The above box presents the report of the world file.

Paragraphs above contain information for other files; Table 5.8 lists the names of the remaining files and lists the sources of information for each.

The author normalized the identity-utility files for this problem so the maximum value of each file is 100. For Table 5.6, the author divided each value by 4. Non-normalized files compete with the utility scaling factor file for weighting the utility on each identity.

File Name	Source of Information
M-60.IDS BRADLEY.IDS T-80.IDS BMP.IDS SA-8.IDS	Table 5.2
M-60.IDU BRADLEY.IDU BMP.IDU SA-8.IDU	Table 5.4 and Table 5.5
T-80.IDU	Table 5.6
EX2.PRR	Table 5.3
EX2.USF	Table 5.7

Table 5.8. Sources of Data for Files in the T/APC/SAM Problem

5.3.2.3 Program Output

```
MostLikelyID = 5

RecommendAction = 1
```

The above box presents the output of the program in Table 5.1, given the above files. The first line of output indicates the target is most likely to have the identity of the fifth-listed identity in the identities file ("SA-8"). The second line of output indicates the action with the highest expected utility is the action with utility values listed first in the identity-utility data files ("shoot an air-to-surface missile").

The output is reasonable. Report "gMftb" the report most likely to occur given that the sensor suite images an SA-8 (Table 5.2). The recommended action has the greatest utility for the SA-8 (Table 5.4). Table 5.9 presents the probabilities the program computed for each identity. Table 5.10 presents the utility values the program computed for each action. Information in these two tables and in similar tables later came from a version of the project code modified to report these values.

M-60	Bradley	T-80	BMP	SA-8
1.1%	16.5%	4.1%	33.1%	45.2%

Table 5.9. Computed Probabilities of Identities for Report "gMftb" for the T/APC/SAM Problem

missile	strafe	bomblets	overfly	avoid
481.1	251.9	424.9	60.6	368.9

Table 5.10. Expected Utilities of Each Action for Report "gMftb" in the T/APC/SAM Problem

Table 5.9 shows that this sensor suite needs much improvement. The output indicates that for the report most likely to occur while imaging an SA-8, the probability that the target is an SA-8 is only 45 percent. Even worse, the system recommends attacking a target with a 17 percent chance of being a Bradley fighting vehicle. The sensor suite needs either stronger reports (numbers in Table 5.2 farther from unity), more reports, or both. The function of this sensor suite, though, is example within this document. It serves well in that respect. Though this thesis will not enter into a discussion of "How much risk of fractricide is acceptable?", it should be a highly unusual circumstance that a commander would accept a 17 percent probability of firing on a friendly vehicle.

This discussion highlights an attractive attribute of utility theory: the numbers the systems need to operate can support effective measures of effectiveness of the

system. For example, the Commander can get a good estimate of the probability of fratricide from the system and adjust utilities or order new sensors to compensate. Utility theory is not capable of eliminating fratricide, but it can measure the risk and thus give the Commander an understanding of the environment. Military forces unavoidably risk fractricide in war; with a system based on utility-theory, commanders can estimate the risk. These estimates can help commanders decide the tough issues, "How much risk of fratricide is too much risk? What decisions are available to manage that risk? What are the other impacts of those decisions?"

5.3.2.4 Variations on the T/APC/SAM Program This section presents several problems related to the T/APC/SAM problem and discusses the significance of the alternate results. The discussion concentrates on results and de-emphasizes presentation of files.

How Does the System Perform With the Sensor Reports Most Likely To Occur While Imaging Other Identities? This section discusses measuring the effectiveness of the identifying portion of the system. Utility values and utility theory are not active in that part of the system.

For report "GmFTB", which is the report most likely to occur while the sensor suite images the M-60, the system correctly identifies the M-60 and recommends overflight.

For report "GmFtB", which is the report most likely to occur while the sensor suite images with the Bradley fighting vehicle, the system finds that the probability of the target being a Bradley is equal to the probability of the target being an M-60 (34.3 percent in each case). It could report either identity; it happens to pick the M-60. It recommends overflight. The recommendation is appropriate because the system concludes that over 68 percent of the time, the target vehicle is friendly.

For report "GmfTB", which is the report most likely to occur while the sensor suite images the T-80, the system correctly identifies the T-80 and recommends a strafing pass.

For report "GmftB", which is the report most likely to occur while the sensor suite images the BMP, the system computes the probabilities for each identity as presented in Table 5.11. It reports the target as a T-80 and recommends a strafing pass.

M-60	Bradley	T-80	BMP	SA-8
10.8%	16.1%	40.4%	32.3%	0.5%

Table 5.11. Computed Probabilities of Identities for Report "GmftB" for the T/APC/SAM Problem

This report shows that the sensor suite is not adequately capable of distinguishing these vehicles. On the report most highly likely to occur while imaging a BMP, the system computes a higher probability for another vehicle. The problem is with the sensor suite, not with the system. The weights in Table 5.2 increase the computed probability of the report being a T-80 for the "G" and "B" reports, but do not increase the probability of the report being a BMP for those reports. Hence, the probability for the T-80 is higher.

Is the System Sensitive to Changes in the Prior Distribution?

Table 5.12 presents the original prior distribution (Table 5.3) and an adjusted prior distribution. The adjustment cuts the expected number of SA-8's by one third and should make the system less likely to identify a target as an SA-8. As expected, with all other information remaining the same as the original problem, the system reduces the probability of seeing an SA-8 and increases the probability of seeing each other identity (Table 5.13). The system now identifies the target as a BMP, though it continues to recommend firing an air-to-surface missile.

Version	M-60	Bradley	T-80	BMP	SA-8
Original	40	20	50	20	3
Adjusted	40	20	50	20	2

Table 5.12. Adjustment to the Assessed Relative Probabilities (Prior Information) for the T/APC/SAM Problem

	M-60	Bradley	T-80	BMP	SA-8
Original	1.1%	16.5%	4.1%	33.1%	45.2%
Adjusted	1.3%	19.5%	4.9%	38.9%	35.4%

Table 5.13. Computed Probabilities of Identities for Report "gMftb" for the T/APC/SAM Problem, Both for the Original Problem and After Adjusting the Priors (Table 5.12)

Is the System Sensitive to Changes in the Utility Scaling Factors?

Table 5.14 presents the original utility scaling factors (Table 5.7) and an adjusted set of utility scaling factors. The adjustment cuts the priority the Commander puts on decisions regarding the SA-8, ranking it lower than the two tanks. The adjustment should make actions associated with the SA-8 less attractive. From Table 5.4, it is clear that for the SA-8, the Commander prefers a missile attack over strafing and over bomblets. From Table 5.9, it is clear that in the original problem, the BMP was the next most likely vehicle. From Table 5.4, it is clear that for the BMP, strafing and bomblets are more highly valued than firing a missile. De-emphasizing the SA-8 via the utility scaling factors could change the recommendation of the system from a missile launch to a strafing pass or a bomblet pass. Between those, the more expectable might be the bomblets, because bomblets get high utility from both identities.

Table 5.15 presents the original utility values (Table 5.10) and the new utility values resulting from solving the original problem with the adjusted utility scaling factors. As suggested, the maximum utility is no longer with the "missile" action;

Version	M-60	Bradley	T-80	BMP	SA-8
Original	5	2	5	1	10
Adjusted	5	2	5	2	3

Table 5.14. Adjustment to Utility Scaling Factors for the T/APC/SAM Problem

it is now with "bomblets". Thus, the system reports the identity as an SA-8, but recommends bomblets because of the utility influence of the second-most-likely vehicle, the BMP.

Version	missile	strafe	bomblets	overfly	avoid
Original	481.1	251.9	424.9	60.6	368.9
Adjusted	165.0	125.5	172.0	29.0	116.0

Table 5.15. Expected Utilities of Each Action for Report "gMftb" for the T/APC/SAM Problem, Both for the Original Problem and with the Adjusted Utility Scaling Factors from Table 5.14

Is the System Sensitive to Changes in the Utilities for a Single Identity? Table 5.16 presents the identity-utility values for the SA-8 from Table 5.4 and presents an adjusted set of values for the SA-8. They are different by reversal of the utilities for the missile and bomblet values. For all other factors being equal, the adjusted values should move some utility value from a recommendation for using a missile to a recommendation for using bomblets. As expected, the expected utility values of the "missile" action loses utility and the "bomblets" action gains utility, as Table 5.17 shows. The system continues to identify the target as an SA-8, but recommends use of bomblets rather than an air-to-surface missile.

Is the System Sensitive to Changes in the Ability of the Sensor Suite to Identify Targets? Table 5.18 presents the original ability of the sensor suite to discriminate the BMP and SA-8 vehicles, and an adjusted ability of the first

	missile	strafe	bomblets	overfly	avoid
Original, SA-8	10	4	8	1	8
Adjusted, SA-8	8	4	10	1	8

Table 5.16. Base Utility and Adjusted Base Utility for the SA-8 and for Each Action in the T/APC/SAM Problem

Version	missile	strafe	bomblets	overfly	avoid
Original	481.1	251.9	424.9	60.6	368.9
Adjusted	390.8	251.9	515.2	60.6	368.9

Table 5.17. Expected Utilities of Each Action for Report "gMftb" for the T/APC/SAM Problem, Both for the Original Problem and with the Adjusted Identity-Utility Information from Table 5.16

sensor for the BMP. In the original sensor suite, the first sensor is equally likely to report the target has a gun ("G") as not ("g"); in the adjusted sensor suite, the first sensor is one quarter as likely to report the target has a gun. The adjusted values bring the most likely report for the BMP closer to the most likely reading for the SA-8. Thus, all other factors being equal, the system might have greater difficulty distinguishing between the two, but may give greater weight to the pair. Table 5.19 presents the resulting probabilities for each identity. As expected, the sum of the probabilities for the SA-8 and the BMP increases, while the difference of those probabilities decreases. The system now identifies the target as a BMP, but continues to recommend using an air-to-surface missile.

Is the System Sensitive to Changes in the Sensor Reports? All the above solutions worked with the sensor report most likely to occur while imaging the SA-8, "gMftb". Many times, however, the sensor suite will generate reports other than those that are most likely. For instance, when looking at an SA-8, Table 5.2 points out that the fifth sensor will indicate "not big" ("b") twice as often as "big"

Target ID	$\frac{P(G)}{P(g)}$	$\frac{P(M)}{P(m)}$	$\frac{P(F)}{P(f)}$	$\frac{P(T)}{P(t)}$	$\frac{P(B)}{P(b)}$
Original BMP	1	1/3	1/2	1/5	1
Adjusted BMP	1/4	1/3	1/2	1/5	1
Original SA-8	1/4	4	1/5	1/8	1/2

Table 5.18. Adjusted and Original Identity-Sensor Data for the T/APC/SAM Problem

	M-60	Bradley	T-80	BMP	SA-8
Original	1.1%	16.5%	4.1%	33.1%	45.2%
	21.7%			78.3%	
Adjusted	0.9%	13.8%	3.4%	44.2%	37.7%
	18.1%			81.9%	

Table 5.19. Computed Probabilities of Identities for Report "gMftb" for the T/APC/SAM Problem, Both for the Original Problem and After Adjusting the Priors (Table 5.12)

("B"). For those times the system images an SA-8 and reports "gMftB", the system should compute a lower probability for the SA-8. Hopefully, the change will not be excessive. Table 5.20 presents the original distribution of identities and the distribution of identities for the newly-mentioned report. As expected, probability shifted from the SA-8. The large increase in probability for the T-80 indicates this reading is closer to the most likely report for the T-80 ("GrfTB") than the original reading. That large increase is a characteristic of this sensor suite, not the system. This discussion again highlights the ability of utility theory to measure the effectiveness of a sensor suite and help decision-makers manage alterations. With that change, the system changes the assessed identity of the target to "BMP" and changes the recommended action to "bomblets". The recommendation of "bomblets" derives from the well balanced probability distribution between the three enemy vehicles (21 percent, 34 percent, and 23 percent, respectively) and the high value of bomblets in all three (Table 5.4), compared to the other attack options.

	M-60	Bradley	T-80	BMP	SA-8
"gMftb"	1.1%	16.5%	4.1%	33.1%	45.2%
"gMftB"	5.6%	16.8%	21.0%	33.6%	23.0%

Table 5.20. Computed Probabilities of Identities for Reports "gMftb" and "gMftB" for the T/APC/SAM Problem

VI. Conclusions and Recommendations

The first section of this chapter expresses conclusions of this project with impact on weapon system design and on management of combat operations. The second section lists recommendations for further research.

6.1 Conclusions

6.1.1 Future Fighting Aircraft Need Decision Support The Air Force will add effectiveness to its future aircraft if it adds an action recommendation system. The pilots of today's aircraft feel challenged in their role of processors of information and would value spending less of their time processing sensor data. Systems based on the ideas in this thesis have the potential for doing some of the processing of sensor data and for making reasonable recommendations to the pilots. Exclusive of policies of combat commanders, the pilot will still have all the autonomy of today's pilot, but will have more valuable information support in the cockpit.

6.1.2 Utility Theory is a Valuable Decision Support Technique Utility theory is a set of mathematical techniques with promise for the decision support systems of future cockpits. Utility theory identifies optimal courses of action based on sensor information, values of various actions, and pre-existing knowledge of the battlefield. By using this set of techniques, tomorrow's fighters and tomorrow's fighting systems (to include weapon systems, support, command elements, etc.) may cause less fratricide and may achieve more consistent application of the commander's rules of engagement.

6.2 Recommendations

6.2.1 Research the Variance on Utility Values a Commander Would Be Likely To Use. Utility theory is useful for identifying differences in value for various actions. Utility theory contributes little if there are small differences in value.

If commanders are likely to routinely put equal or nearly equal value on all targets and on all types of attacks, the pilot may need only a target recognizer. This is equivalent to a commander telling fighter pilots to go to an area and destroy any bridges, fuel trucks, and tanks they see without regard to relative value. For that type of order, utility theory may contribute little, because the pilot should destroy any targets found.

On the other hand, if commanders are likely to put substantially different values on subsets of the target class, utility theory will help. This is equivalent to a commander telling fighter pilots to go to an area with the primary objective of destroying tanks, a secondary objective of destroying fuel trucks, and a tertiary objective of destroying bridges. For that type of order, utility theory may contribute a great deal. By using fuel remaining as an input, the system could be more likely to recommend an attack on a tertiary target late in the mission than early in the mission. Such a characteristic would take into account the probability of finding a higher value target in the remaining search time.

6.2.2 Research How Best to Make Sensor Data Discrete. This thesis assumed all sensor data is discrete; the assumption worked well for the purposes of this thesis. The most effective techniques of converting continuous sensor reports to discrete inputs to a system like the one described here needs research. This research could be related to work like Singstock's (Singstock 91). However, if a sensor exists whose output cannot be preprocessed into discrete inputs, researchers must extend the theory here to accommodate continuous reports.

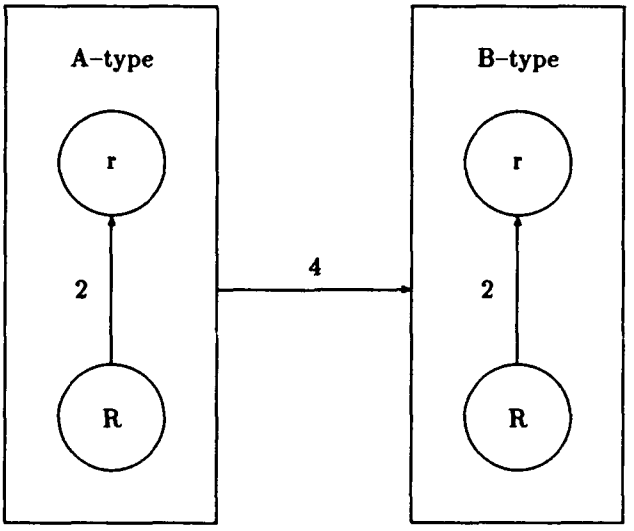
6.2.3 Research Using Pattern Recognition Features for Decision-Making. Some current pattern recognition researchers develop features they feed into neural nets with the goal of classifying the object. For example, Singstock discusses using moments, length-to-width ratios, complexity, compactness, perimeter measures, Fourier transform features, and others (Singstock 91:22). His motivation was to find a set of features useful for characterizing a target as a member of a particular class (in his case, tanks, jeeps, and towers). The idea of the research seems to be that having characterized a target using these features, systems further down the line should get the characterization. Someone should research using these same features as inputs to the decision-making process, rather than simply as inputs to the characterization process. The decision-making process may be capable of making good use of the added information in the features.

6.2.4 Research Giving Prior Distribution Information to the Pattern Recognition System. Given that the intelligence community and the combat operations planners estimate the mix of vehicles the sensors will image, the pattern recognition system may be able to use the estimate (Kabrisky). Many current pattern recognition systems develop estimates without this estimate, the prior distribution.

6.2.5 Improve Input to the Decision-Making System. There are better ways to input data than those the author chose. For example, in the A–type/B–type problem in Chapter 3, three numbers fully specify the eight utility values. That is, Table 3.1 (reproduced in Table 6.1) holds the same information as Figure 6.1. Figure 6.1 is a utility ratio net, different from a probability ratio net only in that the weights on the arcs represent relative utilities instead of relative probabilities. This approach might be easier for users and often would involve less data. The thesis code does not use this technique because it would have added complexity to the thesis code without contributing to the proof-of-concept.

Ordnance	A-type		B-type	
	Not Ready	Ready	Not Ready	Ready
Missile	1	2	8	16
Strafe	2	4	4	8

Table 6.1. Utility Values Computed After Assuming the Lowest Utility is "1"



Symbology: same as Table 3.15

Figure 6.1. Utility Ratio Net for the A-type/B-type Example

6.2.6 Research the Difficulty of Developing the Identity-Sensor Information. The identity-sensor information, as it applies to any one target identity and any sensor suite, is the probability distribution associated with the range of sensor suite reports. That is, for the j possible reports of the sensor suite, it is the set of probabilities $P(Report_j|Identity)$ of each report. If the effort required of the intelligence community to develop this distribution is too high, the ideas in this work will need modification.

6.2.7 Research How Combat Operations Planners Can Best Support a Decision-Recommendation System. Combat operations planners will need to manage development of prior information and utility information for the system proposed here. Many questions come up.

- Can the workload of developing this information be practical?
- Will the resulting combat system be flexible enough?
- What factors will operations planners want as inputs to development of prior information and utility information?
  - Fighter location?
  - Fuel remaining in the fighter's tanks?
  - Combat damage to the fighter?
  - Pilot skill/experience?
  - Time of day or proximity in time to combat planning times, like H-Hour?
  - Pilot preferences?

6.2.8 Research the Pilot Interface. The U.S. Air Force gets much value from training its pilots to be autonomous decision-makers. This system supports the pilot without infringing on the pilot's autonomy. Obviously, the pilot must have inputs.

The range and means of that input needs definition. Further, someone needs to define the information to give to the pilot. This system should not simply add more information to an already crowded cockpit. In discussion of the A-type/B-type problem in paragraph 3.3.2, there is mention of conflict between presenting the most likely identity of the target and the recommended action; such issues are important.

6.2.9 Market Ideas Before Implementation. U.S. Air Force pilots are, as a group, generally proud of their ability to make decisions. There is strong potential of strong resistance to any systems which recommend actions to the pilot. Pilots will fear being "second guessed" and resent the system. Research and development officers will need strong and specific proofs of the value of an action recommendation system before they suggest it to senior commanders.

Appendix A. *The Thesis Code: A User's Guide*

A.1 Using the Thesis Code

To use the thesis code, users create the needed data files and use a short Turbo Pascal program to process them. The author used version 6.0 of the Turbo Pascal compiler. The code has worked with Turbo Pascal version 5.5 without difficulty; the author has not tested compatibility with other versions of the compiler.

Table A.1 is a full user program using the thesis code. The user program gains access to the thesis code via the "USES ProjModU" statement. The thesis code provides a single procedure for use; its name is "ProjMod". Procedure "ProjMod" has six input parameters and two output parameters. Each of the six inputs is a file name. Table A.1 shows extensions in the correct places; the extensions are allowed but the extensions are not required.

The variable "MostLikelyID" returns the number in the list of possible identities of the identity with the highest probability of representing the target. The lowest number returned is 1; the highest number returned is the number of possible identities.

The variable "RecommendAction" returns the number in the list of actions of the action with the highest expected utility. The lowest number returned is 1; the highest number returned is the number of actions.

A.2 Format of Input Files

All input files are MS-DOS text files. Thus, the files have only standard characters in them, except for characters used to mark the end of each line. In particular only characters in the ASCII (American Standard Code for Information Exchange) set may be in the file. With the exception of the possible actions file ("*.PAC"), each data file is a series of fields, with each field delimited by either a

```
{$M 65520,0,655360} { - allows use of maximum memory for both
                      stack and heap }

PROGRAM ThesisCaller;

USES
  ProjModU;

VAR
  MostLikelyID,
  RecommendAction: INTEGER;

BEGIN { - of PROGRAM Caller }

  ProjMod ( 'TEST.PID' , 'TEST.PAC' , 'TEST.ESP' ,
    'TEST.ROW' , 'TEST.PRR' , 'TEST.USF' ,
    MostLikelyID , RecommendAction );
  WRITELN ( 'MostLikelyID = ' , MostLikelyID );
  WRITELN ( 'RecommendAction = ' , RecommendAction );

END.
```

Table A.1. Sufficient Source Code to Use the Thesis Code

space character or an end-of-line. Each field must be in a form readable by Turbo Pascal. Examples are in Table A.2. If the thesis code finds a "%" character, it ignores that character and all characters following it on the same line. Thus, the "%" is useful for placing comments in the input files.

```
1 2 3 4   % first four values
```

```
1  % first value
2  % second value
3  % third value
4  % fourth value
```

Unlike editors, which display the file differently on the screen as a result of finding an end-of-line marker, the thesis code makes no distinction between a space character and an end-of-line marker. Thus, the two boxes above, which are the same except for nments and end-of-line markers, create the same impact on the thesis code.

Example Input Form	Common Representation
2	2
2e3	2 times 10^3; 2000
2.1	2.1
-2.1E+3	-2.1 times 10^3; -2100
2.1E-3	2.1 times 10^{-3}; 0.0021

Table A.2. Example Acceptable Input Forms for Turbo Pascal Numbers (Turbo Pascal:9–10)

A.3 Building the Possible Identities File (".PID")*

The possible identities file has the extension "PID" and is a sequence of fields. Each field is the MS-DOS file name (without an extension) of two files

holding information about one identity. Those two files have the extensions "IDS" and "IDU", respectively; this document discusses them under appropriate headings later. Because of MS-DOS file-naming conventions, no field may be longer than eight characters. The user will probably want to name the files for the identities represented by the data.

```
T-72  % T-72 info is in T-72.IDS and T-72.IDU
T-80  % T-80 info is in T-80.IDS and T-80.IDU
```

If a problem had only two possible identities, the T-72 tank and the T-80 tank, the above box is a useful file.

A.4 Building the Possible Actions File (".PAC")*

The possible actions file has the extension "PAC" and is a sequence of lines. Of the input files for the program, this is the only one the program does not interpret as a series of fields. For purposes of processing, the thesis code determines the number of possible actions by counting the number of lines in the file. Otherwise, the program makes no use of the information in the file, so the contents of the lines is unimportant to processing. The user has the option of using the contents of these lines for any purpose and the option of defining what the contents of these lines should be. In practice, it is likely the user will want a description of the action represented.

```
shoot now with air-to-surface missile
strafe now
turn toward; collect more information
```

The above box is a short representative of an action file.

A.5 Building the Expected Sensor Performance File (".ESP")*

The expected sensor performance file has the extension "ESP" and is a sequence of fields. There are as many fields in the file as there are sensors in the sensor suite. Each field contains the number of distinct reports one sensor reports.

3 2

For a sensor suite with two sensors, the above box might be a useful file. It indicates that the first sensor has 3 distinct reports and that the second sensor has 2 distinct reports. The number of distinct sensor reports possible from a sensor suite is the product of the values in the fields of this file. In the case of this, there are 6 possible reports (3 times 2).

A.6 Building the Report of the World File (".ROW")*

The report of the world file has the extension "ROW" and is a sequence of fields. There are as many fields in the file as there are sensors in the sensor suite. Each field contains the number of the report coming from one sensor. Each field must be within the range of the corresponding value in the expected sensor performance file.

3 1

For the sensor suite described by paragraph A.5, the above box is a valid report. It indicates the first sensor is reporting its third report and the second sensor is reporting its first report.

A.7 Building the Priors File (".PRR")*

The priors file has the extension "PRR" and is a sequence of fields. There are as many fields in the file as there are identities listed in the possible identities file

("*.PID"). Each field is a relative probability; fields are in the same order as the possible identities file. The program derives a probability distribution for the input fields by summing all the fields and dividing each field by that sum.

3 7

For the example possible identity file in paragraph A.3, if the probability of seeing a T-80 is 70 percent while the probability of seeing an T-72 is 30 percent, the above box would be a useful file.

A.8 Building the Utility Scaling Factors File (".USF")*

The utility scaling factors file has the extension "USF" and is a sequence of fields. There are as many fields in the file as there are identities listed in the possible identities file ("*.PID"). Each field is a relative utility; fields are in the same order as the possible identities file. The program uses these numbers to compute utility numbers; this document discusses that computation under the heading "Computing Utility Values" below.

1 2

For the example possible identity file in paragraph A.3, if the commander wants to put twice the weight on actions taken against a T-80 as on actions taken against an T-72, the above box is a useful file.

A.9 Building the Identity-Sensor Data Files (".IDS")*

The identity-sensor data files have the extension "IDS" and are sequences of fields. There are as many identity-sensor data files as there are identities in the possible identities file ("*.PID"). Each identity-sensor data file has one field for each possible sensor report. Each field is the relative probability of one sensor report

relative to all others. The program computes a probability distribution of the reports by summing all the fields and dividing each field by the sum.

The order of the data fields is important. The first values in the file are the values for the first report of the first sensor. The next values in the file are for the second report of the first sensor. The pattern continues through the last report of the first sensor and the end of the file. Within each group of values for reports of the first sensor, the file groups values for reports of each succeeding sensor. Table A.3 demonstrates the order.

first sensor	second sensor	(other sensors) ...	last sensor	fields in file
first report	first report	...	first report	
			⋮	⋮
			last report	
	(other reports)		⋮	⋮
	last report	...	first report	
			⋮	⋮
			last report	
(other reports)	⋮		⋮	⋮
last report	first report	...	first report	
			⋮	⋮
			last report	
	(other reports)		⋮	⋮
	last report	...	first report	
			⋮	⋮
			last report	

After the first sensor, examples of all reports of a sensor are embedded in groups of examples of reports for the next-lower-numbered sensor.

Table A.3. Ordering of Fields in Identity-Sensor Data Files

```
40 10 10 20 10 10
```

The above box is an example identity-sensor data file corresponding to the sensor suite specified in paragraph A.5. It corresponds to one possible identity. It indicates that for that identity, analysts expect the combination of first reports from each sensor 40 percent of the time. Similarly, it indicates analysts expect the combination of second reports from each sensor 20 percent of the time.

A.10 Building the Identity-Utility Data Files (".IDU")*

The identity-utility data files have the extension "IDU" and are sequences of fields. There are as many identity-utility data files as there are identities in the possible identities file ("*.PID"). Each identity-utility data file has one field for each possible sensor report. Each field is the relative utility of one combination of action and sensor report. The program computes a utility value for the combination as discussed below under the heading "Computing Utility Values".

The order of the data fields is important. The first fields in the file are the values for the first action. The next fields in the file are values for the second action. The pattern continues through the last action. Each group of fields for actions is like an identity-sensor data file: there are as many fields as there are distinct reports from the sensor suite and the order of the fields is the same as the identity-sensor data files. Table A.4 demonstrates the order.

```
2 1 1 1 1 1   % first action
3 1 1 1 1 1   % second action
4 1 1 1 1 1   % third action
```

The above box is an example of an identity-utility data file corresponding to the sensor suite specified in paragraph A.5 and to a problem with three possible actions. That table indicates that a large group of combinations of sensor-reports and actions have equal utility. It also indicates that combinations corresponding to the first reports from each sensor and each of the three actions have twice, three times, and four times the utility, respectively, of each member of the large group.

actions	first sensor	(other sensors) ...	last sensor	fields in file
first action	first report	...	first report	
			⋮	⋮
			last report	
	(other reports)		⋮	⋮
	last report	...	first report	
			⋮	⋮
			last report	
(other actions)	⋮		⋮	⋮
last action	first report	...	first report	
			⋮	⋮
			last report	
	(other reports)		⋮	⋮
	last report	...	first report	
			⋮	⋮
			last report	

Table A.4. Ordering of Fields in Identity-Utility Data Files

A.11 Computing Utility Values

The program uses information both from the utility scaling factor ("*.USF") file and the identity-utility data ("*.IDU") files to compute actual utility. For each identity, it multiplies the corresponding factor from the utility scaling factor file by each number of the identity-utility data file to arrive at the utility number it uses.

4	2	2	2	2	2
6	2	2	2	2	2
8	2	2	2	2	2

If the identity-utility data file in paragraph A.10 had a corresponding entry in the utility scaling factor file of "2", the utilities the program computes for that identity are those in the above box.

Appendix B. *Project Code Data Flow Diagrams*

The author used data flow diagrams in designing the project code. He based his techniques on the work of Powers, Adams, and Mills (Powers and others 84). This appendix contains the resulting program design documents.

B.1 Supporting Documents for the Context Diagram

The context diagram depicts relationships of the project code to the world outside its system. Figure B.1 shows the relationships graphically.

B.1.1 Data Dictionary: External Entities This section contains definitions of data flows in Figure B.1.

Intel: a source of intelligence information about the probable distribution of sensor reports to come from the battlefield

Moving Map Display: a display assumed to exist on the production aircraft which depicts, among other things, the assessed identity of known vehicles on the battlefield. Presumably, such a display would also depict terrain and threats to the aircraft, but that portion of this display is outside the scope of this project.

Recommended Action Display: a display assumed to exist on the production aircraft which communicates a recommended course of action to the pilot. This "display" might take the form of audio information for the pilot or, if visual, might be on a heads-up display or on the instrument panel. The form of the display is outside the scope of this project.

Rules of Engagement: for purposes of this system, a set of utility values reflecting assessments made by combat commanders on the value of making various kinds of attacks. For example, "it is twice as valuable to attack an A-type launcher with a strafing pass as with a missile."

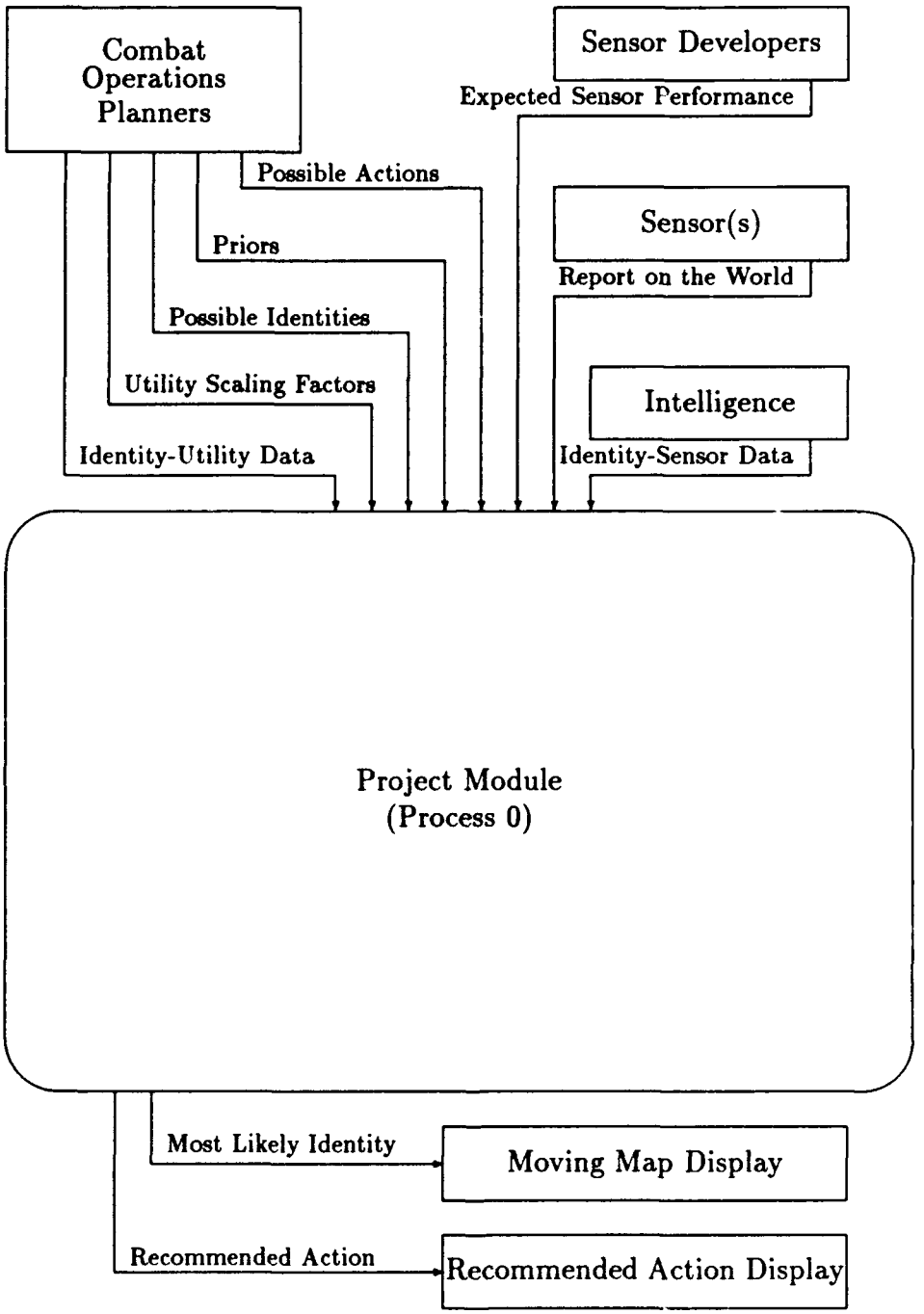


Figure B.1. Context Diagram

Sensor(s): the part of the aircraft system responsible for delivering to the project module a "Report on the World". Sensors can be on board the aircraft (like a radar, a laser, or an infrared camera). Sensors can be off the aircraft, with data reported to the aircraft. For the project module, the difference is not relevant, because the project module is allowed to simply assume existence of a "Report on the World".

Sensor developers: the part of the research and development community responsible for describing the reports to be provided by each sensor

Target Identifier: the part of the aircraft system responsible for (1) identifying a particular potential target as an item of interest, (2) listing possible identities of the item of interest, and (3) listing actions to consider in response to identifying the target of interest

B.1.2 Data Dictionary: Data Items, Logical Definitions This section has definitions of data items intended to communicate the meaning of the data flow, but not the form of the data flow.

Expected Sensor Performance (from "Sensor Developers" to "Project Module"): a description of all information to be communicated from the sensor package to the Project Module

The sensor information is assumed to be discrete (see Thesis, chapter 1). Thus, this piece of data will specify either a list or a range of integer values. Other parts of the Project Module use this information to extract meaning from the sensor data provided.

Identity-Sensor Data (from "Intel" to "Project Module"): the probability distribution of the possible "Report on the World" reports given that the sensor suite is sensing one particular "Possible Identity". This information must be available for each possible identity.

Identity Utility Data (from "Rules of Engagement" to "Project Module"): for one entry in "Possible Identity", a matrix of values, each representing one combination between each entry in "Possible Actions" on one hand and possibilities for "Report on the World" on the other. This information must be available for each possible identity.

Most Likely Identity (from "Project Module" to "Moving Map Display"): the identity from the list "Possible Identities" which has the greatest associated probability of identity

Possible Actions (from "Target Identifier" to "Project Module"): a list of possible responses to identifying the item of interest. This list of actions is assumed to be exhaustive (see Thesis, chapter 1). The Project Module must make a selection from this list to report to "Recommended Action Display".

Possible Identities (from "Target Identifier" to "Project Module"): a list of possible identities of the item of interest. This list of identities is assumed to be exhaustive (see Thesis, chapter 1). The Project Module must make a selection from this list to report to "Moving Map Display".

Priors (from "Intel" to "Project Module"): a set of values chosen by the intelligence community to represent the expected relative proportion of each combination of expected vehicle identity and expected sensor reports

Recommended Action (from "Project Module" to "Recommended Action Display"): the action from the list "Possible Actions" which has the greatest expected utility

Report on the World 'from "Sensor(s)" to "Project Module"): information known about the item of interest, as reported by the sensors

Utility Scaling Factors (from "Rules of Engagement" to "Project Module"): the set of utility values selected by combat commanders to represent the way they want the battle fought

B.1.3 Process Description This section contains the formal definition of the Project Module.

Project Module: the code developed for this project. It takes the inputs defined, uses probability ratio nets to develop assessments for both the identity of the target and the recommended course of action, and reports the results.

B.1.4 General Information About Data Files Used Here The system will accept information in the form of DOS text files. These files will be composed entirely of ASCII text characters, except for carriage returns and line feed characters used to delimit lines. Fields in the files are delimited by either blanks, end-of-line representations, or a combination of any number of the two in any order. In some cases, this documents specifies information to appear alone on a line, in which case the carriage return and line feed characters must form the delimiter.

Case sensitivity. Where possible, input will be case insensitive.

Comments. The file creator may put comments in the file by using the "%" character. Once encountering this character, the program will ignore all remaining text on the line. If the "%" character is the first character on the line (or the first character after leading blanks), the program will ignore the entire line.

Line Length. The program will ignore all characters past the 255th on any one line.

Applicable files. This information applies to files with extensions ".ESP", ".PAC", ".PID", ".IDS", ".IDU", ".ROW", ".PRR", and ".USF".

B.1.5 Data Dictionary: Data Items, Physical Definitions

B.1.5.1 Expected Sensor Performance (ESP) The name of a file created according to the following specification to identify the characteristics of the

sensors. The file must have the extension ".ESP"; the extension is allowable in the input data, but not required.

Format of ".ESP" Files Fields: for each sensor in the sensor suite, the number of distinct reports the sensor is capable of producing. The number of fields is the number of sensors in the suite.

Example: a sensor suite contains two sensors. The first sensor can produce 4 distinct reports; the second sensor can produce 5 distinct reports.

```
4 5
```

The above box is a corresponding ".ESP" file.

B.1.5.2 Possible Actions (PAC) The name of a file created according to the following specifications to list the possible actions the system should consider. The file must have the extension ".PAC"; the extension is allowable in the input data, but not required.

Format of ".PAC" Files Each line: one label for a each possible action. Labels may contain instances of the space character. The number of lines in the file is the number of actions to consider.

Example: A system has three actions to consider with respect to a given target: "attack", "avoid", and "ignore".

```
attack
avoid
ignore
```

The above box could be the ".PAC" file.

B.1.5.3 Possible Identities (PID) The name of a file created according to the following specifications to list the possible identities of the item of interest. The file must have the extension ".PID"; the extension is allowable in the input data, but not required.

Format of ".PID" Files Fields: file names for each possible identity. These labels direct the program to look for files with ".IDS" and ".IDU" extensions (discussed below) for "Identity-Sensor Data" and "Identity Utility Data". Hence, the label can be no longer than eight characters (MS-DOS limit of file names) and must use only characters acceptable in MS-DOS file names. MS-DOS accepts the following characters: A..Z, 0..9, and members of the string

_^$~!#%&-{}()@'`

The number of fields in the file is the number of identities.

M-60 T-72

Example: If an item of interest is known to be either an "M-60" tank or a "T-72" tank, the ".PID" file in the above box could be appropriate. This file directs the program to look for files "M-60.IDS" and "T-72.IDS" for data on detection by the sensor suite, and "M-60.IDU" and "T-72.IDU" for data on utilities.

B.1.5.4 Identity-Sensor Data (IDS) Information on the relative frequency of each possible "Report on the World" report, given that the sensor suite is gathering data on a particular identity of target. The program reads the information from a disk file which must have the extension ".IDS". The file title should identify the particular identity. The program obtains the first part of the file title from "Possible Identities" and adds the ".IDS" extension.

Format of ".IDS" Files There will be as many values in the file as the number of distinct "Report on the World" reports that are possible. That number of reports is the product of

- the number of distinct reports possible from the first sensor,
- the number of distinct reports possible from the second sensor, . . . and
- the number of distinct reports possible from the last sensor.

Each value in the file must be in form for Turbo Pascal to interpret as a floating point number (including integer values). Each value represents the relative probability of one "Report on the World" report, given that the target is of the type identified in the file title.

The first group of values applies to the first report the first sensor can produce, as listed in "Expected Sensor Performance". The next group is for the second report of the first sensor, etc., continuing to the last report of the first sensor.

Within each of those groups, the first group of values applies to the first report the second sensor can produce, as listed in "Expected Sensor Performance". The next group is for the second report the second sensor can produce, etc.

This hierarchical pattern continues until having reached the last sensor.

The program converts the relative probability information in the file to a probability distribution by dividing each value in the file by the sum of all values.

Example: Some problem uses a sensor suite with 2 sensors. One sensor reports one of two items from a list; the other sensor reports one of three integers from a range.

```
2 3 % two sensors; first has 2 reports; other has 3
```

The above box could be the ".ESP" file.

```
4  % sensor 1, report 1; sensor 2, report 1
4  % sensor 1, report 1; sensor 2, report 2
2  % sensor 1, report 1; sensor 2, report 3
2  % sensor 1, report 2; sensor 2, report 1
5  % sensor 1, report 2; sensor 2, report 2
3  % sensor 1, report 2; sensor 2, report 3
```

This sensor suite can produce only six distinct "Report on the World" reports (the product of 2 reports and 3 reports). For this problem and for some identity relevant to the problem, the file in the above box could be appropriate.

The sum of those six numbers is 20, so the probability of the first report is 4/20, or 0.2. Thus, the distribution for this file is the set of values, (.20, .20, .10, .10, .25, .15). The program would accept those values directly with the same result.

B.1.5.5 Identity Utility Data (IDU) A set of files, each corresponding to one entry in "Possible Identities". Each file contains a relative utility of every combination of entries in "Possible Actions" and possible "Report on the World" reports. The program reads the information from a disk file which must have the extension ".IDU". The file title should identify the particular identity. The program obtains the first part of the file title from "Possible Identities" and adds the ".IDU" extension.

Format of ".IDU" Files The file will have as many relative utility numbers as the product of the number of distinct reports specified in "Expected Sensor Performance" and the number of entries in "Possible Actions". Each relative utility must be in a form that Turbo Pascal can interpret as a floating point number (including integers).

The first group of utility values applies to the first report of the first sensor as listed in "Expected Sensor Performance". The next group of utility values applies

to the second report of the first sensor, etc., continuing until the last report of the first sensor.

Within each of those groups, the first group of utility values applies to the first report of the second sensor as listed in "Expected Sensor Performance". The next group of utility values applies to the second report of the second sensor, etc., continuing until the last report of the second sensor.

This repeating hierarchy of sensors continues until reaching the last sensor.

Within each of those groups, the first value applies to the first listed action in "Possible Actions". The second value applies to the second listed actions, etc., continuing to the last listed action.

```
fire a missile
strafe
```

Example: On a given problem, "Possible Actions" includes 2 entries: "fire a missile" and "strafe". The above box could be the possible actions file.

```
2  % first sensor, 2 reports
3  % second sensor, 3 reports
```

The sensor suite includes one sensor that reports "has treads" or "has tires" (2 reports), and one sensor that reports an integer, either "3", "4", or "5" (3 reports). The above box could be the expected sensor performance file.

```
      % values for first sensor:  "has tires"
        % values for second sensor:  "1"
 1  2     % values for "fire a missile" and "strafe"
        % values for second sensor:  "2"
 3  4     % values for "fire a missile" and "strafe"
        % values for second sensor:  "3"
 5  6     % values for "fire a missile" and "strafe"
      % values for first sensor:  "has treads"
        % values for second sensor:  "1"
 7  8     % values for "fire a missile" and "strafe"
        % values for second sensor:  "2"
 9 10     % values for "fire a missile" and "strafe"
        % values for second sensor:  "3"
11 12     % values for "fire a missile" and "strafe"
```

For any particular target identity, this ".IDU" file must have 12 fields (2 reports on the first sensor times 3 reports on the second sensor times 2 actions). The above box could be appropriate.

These relative utilities correspond to only one entry in "Possible Identities"; similar information must be available for all entries.

B.1.5.6 Priors (PRR) The name of a file created according to the following specifications to list the probability distribution expected to exist between the identities listed in "Possible Identities". The file must have the extension ".PRR"; the extension is allowable in the input data, but not required.

Format of ".PRR" Files This file holds the relative probability of each of the identities listed in "Possible Identities" and in the same order. The values in the file must be in a form Turbo Pascal can interpret as a REAL value

(including INTEGER values). After reading the whole file, the program will form the probability distribution it needs by dividing each value by the sum of all values.

```
2  % relative probability of the first identity
1  % second identity
1  % third identity
```

Example: in some problem, there are three identities. The first one listed is assumed to be twice as probable as each of the others. The above box could be appropriate.

The sum of those numbers is 4, so the program computes the probability of the first identity as 2/4, or 0.5. Thus, the probability distribution for this file will be the vector (.50, .25, .25).

B.1.5.7 Report on the World (ROW) The name of a file created according to the following specifications to specify the possible sensor reports "Report on the World". The file must have the extension ".ROW"; the extension is allowable in the input data, but not required.

Format of ".ROW" Files This file holds the report of the each sensor in the suite on the item of interest. If the sensor reports an integer reading, the difference between the reading and the low value of the reporting range is here. If the sensor reports an item from a list, the integer is the one corresponding to the appropriate label. The first item on the list is item number 1. Thus, for both types of sensors, this file records an index to the sensor's allowable range, as listed in "Expected Sensor Performance". Sensor reports are in the same order in this file as the sensor descriptions in "Expected Sensor Performance".

```
2 3  % 2 sensors:  first with 2 reports, other with 3
```

Example: A sensor suite S has two sensors. One produces 2 reports and one produces 3 reports. The above box could be appropriate.

```
2  % first sensor
3  % second sensor
```

In sensor suite S, the first sensor reports its second-listed report and the second sensor reports its last-listed report. The above box could be the corresponding ".ROW" file.

B.1.5.8 Utility Scaling Factors (USF) The name of a file created according to the following specifications to list the scaling factors the program should use on values in "Identity Utility Data" to create utility values. The file must have the extension ".USF"; the extension is allowable in the input data, but not required.

Format of ".USF" Files This file holds a utility scaling value for each of the identities in "Possible Identities", listed in the same order as that list. The value in the file must be in a form Turbo Pascal can interpret as a REAL variable (including integers). The program will compute utility for a combination of values of "Possible Action" and "Possible Identities" as the product of values in this file and the utility value from "Identity Utility Data".

```
2  % relative utility of first action on identity ID1
1  % second action on ID1
```

```
3  % relative utility of first action of identity ID2
4  % second action on ID2
```

Example: a particular problem has 2 entries in "Possible Action" and 2 entries in "Possible Identities". The above boxes could be appropriate as utility files—one for each identity.

```
6  % first identity
5  % second identity
```

The above box could be appropriate as a utility scaling factor file.

From the above three files, the program will compute the utility of "first identity, first action" as 6 X 2 = 12. Similarly, the program will compute the utility of "second identity, second action" as 5 X 4 = 20.

B.1.5.9 Most Likely Identity An integer identifying the position on the list of "Possible Identities" which the target is most likely to be. The first item on the list is item 1.

```
M60  T72
```

Example: Given a "Possible Identities" with two items, M-60 and T-72, the above box could be appropriate.

Given also that the target is most likely to be a T72, "Most Likely Identity" would contain the integer "2".

B.1.5.10 Recommended Action An integer identifying the position on the list of "Possible Actions" which has the highest expected utility.

```
strafe
fire missile
```

Example: Given a "Possible Actions" with two actions, "strafe" and "fire missile", the above box could be appropriate.

Given also that the recommended action is "strafe", "Recommended Action" would contain the integer "1".

B.1.6 Computing Platform This project will be coded and specified for the MS-DOS Turbo Pascal programming environment, assumed to be running on a machine with at least 640K of basic memory and available external storage (such as, a hard disk).

B.2 Project Code Diagram 0

Within the context diagram, above, is a process numbered "0". Diagram 0 depicts relationships of parts of that process. Figure B.2 depicts the relationships graphically.

B.2.1 Data Dictionary: Data Items, Logical Definitions This section contains definitions of data items intended to communicate the meaning of the data flow, though not the form of the data flow.

Distribution of Identities (from "Identifying Process" to "Action Assessor"): a vector with "Number of Identities" elements, each of which represents the probability of its corresponding identity given the "Report on the World". The elements of the vector form a discrete probability distribution, so their sum is 1.

Expected Sensor Performance (from external to "Process Starter"): (full definition at Context Diagram level)

Identity-Sensor Data (from external to "Process Starter"): (full definition at Context Diagram level)

Identity-Utility Data (from external to "Process Starter"): (full definition at Context Diagram level)

Local Priors (from "Process Starter" to "Identifying Process"): a probability ratio net, using the data structures provided by the PNET UNIT, holding the probability information in "Priors" and "Identity-Sensor Data" with nodes labeled to support appropriate grouping actions in "Identifying Process"

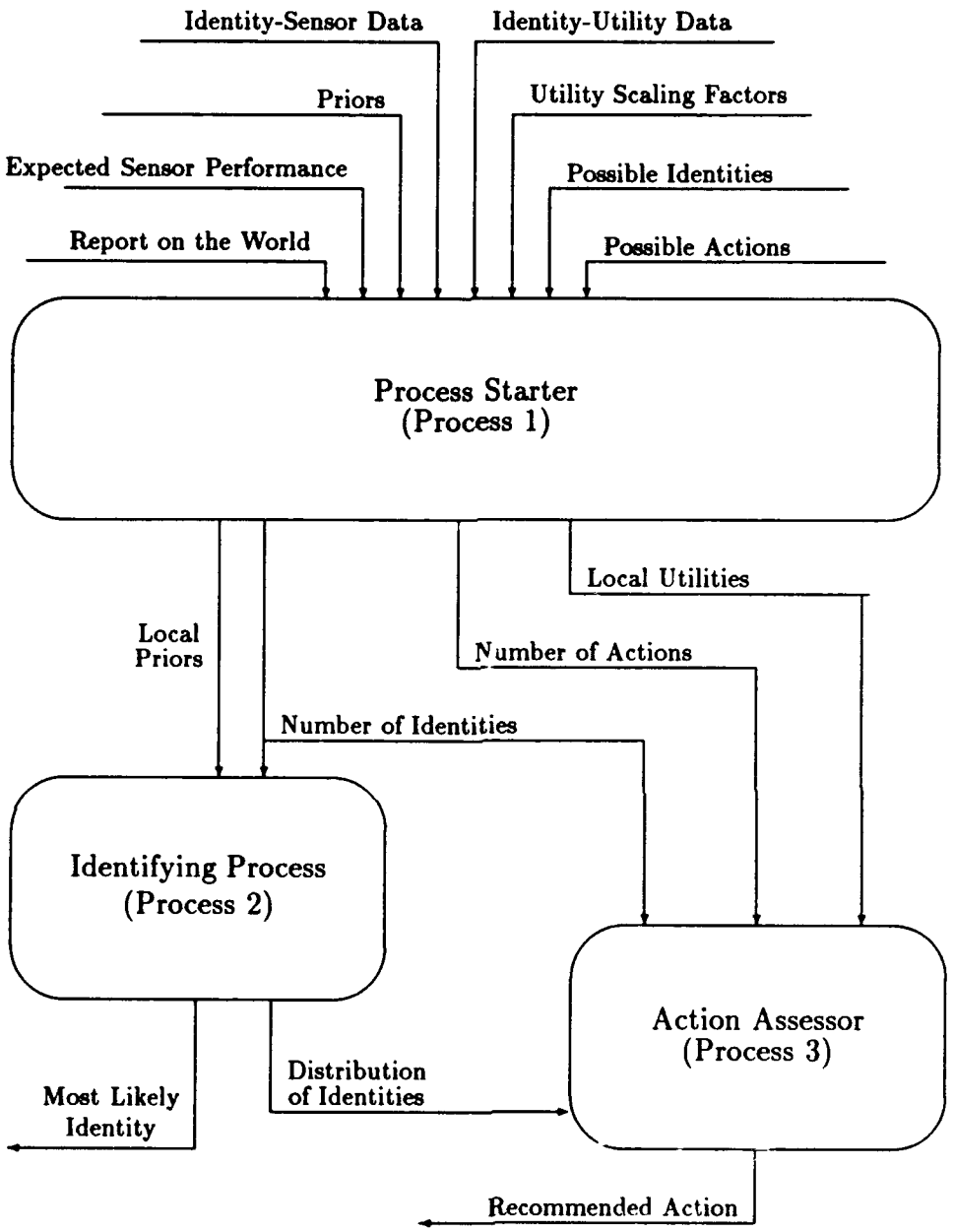


Figure B.2. Diagram 0

Local Utilities (from "Process Starter" to "Action Assessor"): an internal representation of the needed information in "Identity-Utility Data" and "Utility Scaling Factors". Only information relevant to "Report on the World" is present.

Most Likely Identity (from "Identifying Process" to external): (full definition at Context Diagram level)

Number of Actions (from "Process Starter" to "Action Assessor"): the number of actions received in "Possible Actions"

Number of Identities (from "Process Starter" to "Identifying Process" and "Action Assessor"): the number of identities received in "Possible Identities"

Possible Actions (from external to "Process Starter"): (full definition at Context Diagram level)

Possible Identities (from external to "Process Starter"): (full definition at Context Diagram level)

Priors (from external to "Process Starter"): (full definition at Context Diagram level)

Recommended Action (from "Action Assessor" to external): (full definition at Context Diagram level)

Report on the World (from external to "Process Starter" and "Identifying Process"): (full definition at Context Diagram level)

Utility Scaling Factors (from external to "Process Starter"): (full definition at Context Diagram level)

B.2.2 Data Dictionary: Process Descriptions This section contains formal descriptions of the three major processes.

Action Assessor (Process 3): performs vector inner product with "Distribution of Identities" and selected parts of "Local Utilities" to compute a vector of expected

utility values, one for listing in "Possible Actions". Reports the position of the maximum element of the vector as "Recommended Action".

Identifying Process (Process 2): uses tools provided by the PRNTHRM UNIT to convert the probability ratio net "Local Priors" to a form that gives a vector "Distribution of Identities". Reports the position of the maximum element of that vector as "Most Likely Identity".

Process Starter (Process 1): receives the eight files specified in the Context Diagram document and converts their form into internal form for use of the remainder of the program. The four outputs ("Local Priors", "Local Utilities", "Number of Identities", and "Number of Actions") contain all the information needed from the eight inputs.

B.2.3 General Data Specifications This section lists general data specifications.

The code will dimension the ARRAY's using CONSTANT's.

Data limits:

- For this thesis, the program will support up to 10 sensors.
- Any sensor may have up to 256 reports.
- The maximum number of distinct reports from the sensor suite (the product across sensors of the maximum number of reports of each sensor) is 65535.
- For both "Possible Actions" and "Possible Identities", there must be at least 2 entries and there must be no more than 80 entries.
  - These limits are arbitrary, but deemed both large enough for this thesis and small enough to fit within the capability of the computing platform to be used for this project.

The code will declare INTEGER variables with the range of values expected to be stored there. Thus, the compiler can size the variables to minimum size and save memory.

B.2.4 Data Dictionary: Data Items, Physical Definitions This section contains definitions of data flows intended to establish the form of the data flow.

Distribution of Identities: a REAL array, the elements of which sum to 1.0. Each element represents the computed probability of the corresponding identity, given "Report on the World".

Maximum number of REAL variables used: number of identities. Worst case given above limits: 80 REAL variables. Each REAL variable will use six bytes, for a total use of 480 bytes.

Expected Sensor Performance: (full definition at Context Diagram level)

Identity-Sensor Data: (full definition at Context Diagram level)

Identity-Utility Data: (full definition at Context Diagram level)

Local Priors: a probability ratio net (PRN) stored in the data structures of the PRNTHRM UNIT (Arcs and Nodes), with the structure specified below and with node labels as specified below. The nodes of the PRN represent combinations of "Possible Identity" and "Report on the World".

PRN structure: The PRN will have two levels of hierarchy. At the first level will be nodes representing information for one identity each, in particular the identities listed in "Possible Identities". Embedded within each level will be a PRN with two nodes to represent the relative probability of "Report on the World" against all other possible "Report on the World" reports grouped together.

Node labels: character strings of length 0 or 1

- First level of hierarchy: If "IDNum" holds the 0-based index of the identity, the string will have only ASCII character number "IDNum" (CHR(IDNum)).

Thus, if the first character is "CHR(0)", the node applies to the first identity listed in the PID file and the reported ROW. Similarly, "CHR(1)" indicates the second-listed identity.

- Second level of hierarchy
  - Nodes representing "Report on the World": same as first level of hierarchy
    - There is no ambiguity between these labels and the parent's because the nets will be flattened, a process which includes discarding the node formerly serving as the parent.
  - Nodes representing all other possibilities for "Report on the World": empty. Parts of the program will check for this and interpret it as an order to ignore the node for much subsequent processing.

Maximum number of nodes: three nodes per identity. Worst case given the above limits: 240 nodes. Each node will use 11 bytes for the string variable and 1 byte for the number of the embedded arc, if any. Total use of memory for nodes: 2880 bytes.

Maximum number of arcs: one less than twice the number of identities. Worst case given the above limits: 159 arcs. Each arc will use 1 byte for the number of the node at the head, 1 byte for the number of the node at the tail, and 6 bytes for the associated weight. Total use of memory for arcs: 1272 bytes.

Local Utilities: an REAL array holding the needed utility information from "Utility Scaling Factors" and "Identity-Utility Data". Only the information relevant to "Report on the World" is present.

Data order: The first part of file is for the first action listed in "Possible Actions", then other actions. Within each of those are values for each identity, in order of listing in "Possible Identities".

Maximum number of REAL variables used: product of number of "Possible Actions" and number of "Possible Identities". Worst case number of REAL variables given above limits: 80 X 80 = 6400 REAL variables. Each of these REAL variables will use 6 bytes for total use of 38400 bytes.

Most Likely Identity: (full definition at Context Diagram level)

Number of Actions: the number of actions specified in "Possible Actions", an INTEGER variable

Number of Identities: the number of identities received in "Possible Identities", an INTEGER variable

Possible Actions: (full definition at Context Diagram level)

Possible Identities: (full definition at Context Diagram level)

Priors: (full definition at Context Diagram level)

Recommended Action: (full definition at Context Diagram level)

Report on the World: (full definition at Context Diagram level)

Utilities: (full definition at Context Diagram level)

B.2.5 Check on Feasibility for Data Size Turbo Pascal has a data limit of 65,536 (64K) bytes. By the above computations, the data will require at most 480 + 2880 + 1272 + 38400 bytes = 43032 bytes. Conclusion: the design is feasible; there is adequate space in the data stack for planned data and some more.

Appendix C. *Project Source Code*

This appendix contains the source code for three Turbo Pascal units. The unit called PRNTHRM contains code used to implement Morlan's probability ratio net theorems. The unit called PROJPRST contains code for one procedure of the project code. The unit called PROJMODU is the unit the user will use.

The code in PROJPRST could very well reside in PROJMODU, but the resulting file would exceed the 64K size limit for the editor in Turbo Pascal 5.5 and earlier.

C.1 Source Code for the PRNTHRM Unit

```
{ - VERSION:  Final }

UNIT PrnThrm;

{ - This UNIT holds the theorems from Bruce Morlan's
    dissertation on probability ratio nets.  Use it to
    implement his routines. }

INTERFACE { ------------------------------------------------- }

CONST
  LongestNodeLabel = 10;          { - sizes the data structures }
  MaxNumArcs = 159;               { - sizes the data structures }
  MaxNumNodes = 240;              { - sizes the data structures }
  NoEmbedArc = 0;                 { - ( Arc number = 0 ) means
                                      "this arc has no embedded
                                      arcs" }
  LowArcNum = 1;
  LowNodeNum = 1;

TYPE
  TArcNum = LowArcNum .. MaxNumArcs;
  TNodeNum = LowNodeNum .. MaxNumNodes;
  TArcNode = ARRAY [ TArcNum ] OF TNodeNum;
  TArcREAL = ARRAY [ TArcNum ] OF REAL;
  TLabelWidth = 2 .. 4;
  TNArcs = 0 .. MaxNumArcs;
  TArc =
    RECORD
      NArcs: TNArcs;              { - number of arcs
                                      in current net }
      ArcNumLabelWidth: TLabelWidth;
                                  { - 2 if NArcs < 100, else
```

```
                                    3 if NArcs < 1000, else 4 }
    Head,                          { - node number of head }
    Tail: TArcNode;                { - node number of tail }
    Weight: TArcREAL;              { - probability of head /
                                    probability of tail }
  END; { - TArcList RECORD }
TEmbedNum = NoEmbedArc .. MaxNumArcs;
TSingleNodeStr = STRING [ LongestNodeLabel ];
TNodeStr = ARRAY [ TNodeNum ] OF TSingleNodeStr;
TNodeEmbed = ARRAY [ TNodeNum ] OF TEmbedNum;
TNNodes = 0 .. MaxNumNodes;
TNode =
  RECORD
    NNodes: TNNodes;               { - number of nodes
                                    in current net }
    NodeNumLabelWidth: TLabelWidth;
                                   { - 2 if NNodes < 100, else
                                    3 if NNodes < 1000,
                                    else 4 }
    EmbedArc: TNodeEmbed;          { - pointers to embedded arcs;
                                    0 means "none embedded" }
    NodeLabel: TNodeStr;           { - labels for nodes }
  END; { - TNodeList RECORD }
TStoreArcs = ARRAY [ TArcNum ] OF TArcNum;
TStoreNodes = ARRAY [ TArcNum ] OF TNodeNum;



PROCEDURE Condition
( ArcToEmbed,
  ReplaceArc: TArcNum;
  NewNodeLabel: TSingleNodeStr;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode );

PROCEDURE DirectArcsToNode
( LinkNode: TNodeNum;
  VAR NArcToPoint: INTEGER;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode );

PROCEDURE FlattenPRN
( VAR Arc: TArc;
  VAR Node: TNode;
  VAR ErrorExit: BOOLEAN );

PROCEDURE FormTheJoint
( PenArc: TArcNum;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode );

PROCEDURE Triangulate
( ReplArc,
```

```
    HoldArc: TArcNum;
    VAR ErrorExit: BOOLEAN;
    VAR Arc: TArc );

  PROCEDURE PurgeNodes
  ( VAR Arc: TArc;
    VAR Node: TNode );

  PROCEDURE ReverseArc
  ( ArcToRev : TArcNum;
    VAR Arc: TArc );

  PROCEDURE ScratchNode
  ( NodeToScr: TNodeNum;
    VAR Arc: TArc;
    VAR Node: TNode );

  FUNCTION SumOfWeights
  ( NodeToSumAt: TNodeNum;
    VAR ErrorExit: BOOLEAN;
    VAR Arc: TArc;
    VAR Node: TNode ): REAL;


IMPLEMENTATION { ------------------------------------------- }

  { ----- PROCEDURE Condition -----
    - Input:
      -- Parameters:
         --- ArcToEmbed:  the number of an arc to be embedded
             by this operation
         --- ReplaceArc:  the number of the arc to be replaced by
             this operation
         --- NewNodeLabel:  the label to use for the new node
         --- Arc:  arc information
         --- Node:  node information
      -- Variables:  none
      -- I/O:  none
    - Action:  performs conditioning (creates hierarchy in
      the net)
    - Output:
      -- Parameters:
         --- Arc:  updated arc information
         --- Node:  updated node information
      -- Variables:  none
      -- I/O:  none
    - Screen mode:  text only
    - External subroutines used:
      -- PressAnyKey
      -- ReverseArc
      -- SumOfWeights
    - Last modified:  20 Jan 92 }

  PROCEDURE Condition
```

```
( ArcToEmbed,
  ReplaceArc: TArcNum;
  NewNodeLabel: TSingleNodeStr;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode );

VAR
  WeightFactor: REAL;

BEGIN { - PROCEDURE Condition }

  { - make room for the new node }
  INC (   Node . NNodes   );

  { - change the head of ReplaceArc }
  Arc . Head [ ReplaceArc ]   :=   Node . NNodes;

  { - update the new node }
  Node . EmbedArc [   Node . NNodes   ]   := ArcToEmbed;
  Node . NodeLabel [   Node . NNodes   ]   := NewNodeLabel;

  WeightFactor :=   1 +
    SumOfWeights (   Arc . Head [ ArcToEmbed ]   , ErrorExit ,
      Arc , Node );
  Arc . Weight [ ReplaceArc ]   :=
    Arc . Weight [ ReplaceArc ]   / WeightFactor;

END; { - PROCEDURE Condition }



{ ----- PROCEDURE DirectArcsToNode -----
  - Input:
    -- Parameters:
       --- LinkNode:  the node toward which to direct arcs
           are report back with a list of arcs
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:
    -- puts LinkNode at the head of all arcs it is in
    -- for any arc pointing at the tail of an arc pointing
       to LinkNode, triangulates.  Result:  those arcs
       point at LinkNode, too.
    -- reports back with
       --- how many arcs point to LinkNode
       --- which arcs those are
       --- which nodes are the tails of those arcs
  - Output:
    -- Parameters:
       --- NArcToPoint:  the number of arcs pointing
           to LinkNode at termination
       --- ErrorExit:  a BOOLEAN variable indicating the
           graphics mode was turned off due to an error
```

```
      --- Arc:  updated arc information
      --- Node:  updated node information
    -- Variables:  none
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:
    -- ReverseArc
    -- Triangulate
  - Last modified:  20 Jan 92 }

PROCEDURE DirectArcsToNode
( LinkNode: TNodeNum;
  VAR NArcToPoint: INTEGER;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode );

TYPE
  TArcToPt =
    RECORD
      NodeInArcToPt: TStoreNodes; { - for any arc with one end
                                      at LinkNode, this stores
                                      the other node number }
      ArcNumToPt: TStoreArcs;     { - stores the number of each
                                      arc with an endpoint at
                                      LinkNode }
    END; { - TArcToPt RECORD }

VAR
  OneOfArcsToLN,
  OneOfAllArcs: TArcNum;
  ArcToLN: TArcToPt;

BEGIN { - PROCEDURE DirectArcsToNode }

  WITH ArcToLN DO
    BEGIN

      { - assure that all arcs containing LinkNode point to it;
          count how many point to it; record them }
      NArcToPoint := 0;
      FOR OneOfAllArcs := LowArcNum TO   Arc . NArcs   DO
        BEGIN

          { - reverse arcs with LinkNode at tail }
          IF (   Arc . Tail [ OneOfAllArcs ]   = LinkNode )
          THEN
            ReverseArc( OneOfAllArcs , Arc );

          { - record arcs with LinkNode at head }
          IF (   Arc . Head [ OneOfAllArcs ]   = LinkNode )
          THEN
            BEGIN

              INC ( NArcToPoint );
              NodeInArcToPt [ NArcToPoint ] :=
```

```
                Arc . Tail [ OneOfAllArcs ];
              ArcNumToPt [ NArcToPoint ] := OneOfAllArcs;

          END; { - IF ( Head [ OneOfAllArcs ] = LinkNode }

      END; { - FOR OneOfAllArcs }

    { - find arcs that share a node with an arc pointing to
        LinkNode; use Triangulate to point those arcs at
        LinkNode }
    OneOfArcsToLN := 1; { - initialize }
    WHILE ( OneOfArcsToLN <= NArcToPoint ) DO
      BEGIN
        FOR OneOfAllArcs := 1 TO   Arc . NArcs   DO
          BEGIN

            { - reverse any arcs sharing a tail with an
                arc pointing to LinkNode }
            IF (   Arc . Tail [ OneOfAllArcs ]   =
                 NodeInArcToPt [ OneOfArcsToLN ] ) AND
               (   Arc . Head [ OneOfAllArcs ]   <>
                 LinkNode )
            THEN
              ReverseArc ( OneOfAllArcs , Arc );

            { - for any arcs pointing at the tail of an arc
                pointing at LinkNode, triangulate to make the
                arcs point to LinkNode; record them }
            IF (   Arc . Head [ OneOfAllArcs ]   =
                 NodeInArcToPt [ OneOfArcsToLN ] )
            THEN
              BEGIN

                Triangulate ( OneOfAllArcs ,
                  ArcNumToPt [ OneOfArcsToLN ]  ,
                  ErrorExit , Arc );

                IF ( ErrorExit )
                THEN
                  EXIT;   { - immediate termination
                            of DirectArcsToNode }

                { - record }
                INC ( NArcToPoint );
                NodeInArcToPt [ NArcToPoint ] :=
                  Arc . Tail [ OneOfAllArcs ];
                ArcNumToPt [ NArcToPoint ] := OneOfAllArcs;

              END; { - IF ( H [ OneOfAllArcs ] =
                          T [ OneOfAllArcs ] }
          END; { - FOR OneOfAllArcs }

        INC ( OneOfArcsToLN );
      END; { - WHILE ( OneOfArcsToLN <= NArcToPoint ) }
  END; { - WITH ArcToLN }
END; { - PROCEDURE DirectArcsToNode }
```

```
{ ----- PROCEDURE FlattenPRN -----
  - Input:
    -- Parameters:
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:  completely flattens the probability ratio net
    -- Goes through the arc list one arc at a time.
    -- Checks the head of the arc for an embedded arc.
       Penetrates the embedded arc, if it is there.
       --- Can't be certain that the head is not now embedded;
           the new node could have had an embedded arc at the
           next level of hierarchy, so rechecks.
    -- Checks the tail of the arc for an embedded arc.
       If there:
       --- Calls ReverseArc to put the embedded arc at the
           head.
       --- Penetrates the embedded arc.
       --- Can't be certain that the head is not now embedded;
           rechecks.
    -- If neither head nor tail have embedded arcs, moves to
       next arc.
  - Output:
    -- Parameters:
       --- Arc:  updated arc information
       --- Node:  updated node information
       --- ErrorExit:  indicates a problem in executing the
           routine
    -- Variables:
       --- ErrorExit:  indicates a failure of the effort to
           execute the routine
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:
    -- FormTheJoint
  - Last modified:  20 Jan 92 }

PROCEDURE FlattenPRN
( VAR Arc: TArc;
  VAR Node: TNode;
  VAR ErrorExit: BOOLEAN );

VAR
  LocArc: TArcNum;
  LocArcHeadEmbedArc,
  LocArcTailEmbedArc: TEmbedNum;

BEGIN { - PROCEDURE FlattenPRN }
  LocArc := 1;
  WHILE ( LocArc <=   Arc . NArcs ) DO
    IF (   Node . EmbedArc [   Arc . Head [ LocArc ]   ]   =
         NoEmbedArc )
    THEN
```

```
      IF (   Node . EmbedArc [   Arc . Tail [ LocArc ]   ]   =
           NoEmbedArc )
      THEN
        INC ( LocArc )
        { - done with LocArc:  neither head nor tail
            include embedded arcs }
      ELSE { - tail has an embedded arc }
        BEGIN

          FormTheJoint ( LocArc , ErrorExit , Arc , Node );
          IF ( ErrorExit ) THEN
            EXIT; { - PROCEDURE FlattenPRN }

        END { - LocArcTailEmbedArc <> NoEmbedArc }
    ELSE { - LocArcHeadEmbedArc <> NoEmbedArc }
      BEGIN

        FormTheJoint ( LocArc , ErrorExit , Arc , Node );
        IF ( ErrorExit ) THEN
          EXIT; { - PROCEDURE FlattenPRN }

      END; { - ( LocArcHeadEmbedArc <> NoEmbedArc }
END; { - PROCEDURE FlattenPRN }



{ ----- PROCEDURE FormTheJoint -----
  - Input:
    -- Parameters:
       --- PenArc:  an arc number with an embedded net
           at the head
       --- LinkNode:  a node number in the embedded net at
           which PenArc should point after completion
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:  performs Morlan's basic operation of "forming
    the joint distribution"
  - Output:
    -- Parameters:
       --- ErrorExit:  a BOOLEAN variable indicating
           inability to perform DirectArcsToNode
       --- Arc:  updated arc information
       --- Node:  updated node information
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- DirectArcsToNode
    -- PurgeNodes
    -- SumOfWeights
  - Last modified:  20 Jan 92 }

PROCEDURE FormTheJoint
( PenArc: TArcNum;
  VAR ErrorExit: BOOLEAN;
```

```
  VAR Arc: TArc;
  VAR Node: TNode );

VAR
  LinkNode: TNodeNum;
  LinkArc: TEmbedNum;
  WtFactor: REAL;
  NumArcsPoint: INTEGER;

BEGIN { - PROCEDURE FormTheJoint }

  { - decide on a LinkNode }
  LinkArc :=   Node . EmbedArc [   Arc . Head [ PenArc ]   ];
  IF ( LinkArc = NoEmbedArc )
  THEN
    BEGIN
      LinkArc :=
        Node . EmbedArc [   Arc . Tail [ PenArc ]   ];
      IF ( LinkArc = NoEmbedArc )
      THEN
        { - there is no embedded arc }
        EXIT; { - from FormTheJoint }
      ReverseArc ( PenArc , Arc );
    END; { - IF ( LinkArc = NoEmbedArc ) }
  LinkNode :=   Arc . Head [ LinkArc ];

  { - direct all arcs possible to the tail of the
      penetrating arc.  This precludes having more than
      one arc point to LinkNode, which causes problems. }
  DirectArcsToNode (   Arc . Tail [ PenArc ]   ,
    NumArcsPoint , ErrorExit , Arc , Node );

  IF ( ErrorExit )
  THEN
    EXIT; { - from FormTheJoint }

  { - An unwanted byproduct of DirectArcsToNode
      is the reversal of PenArc.  Reverse it back. }
  ReverseArc ( PenArc , Arc );

  { - update the penetrating arc as required by Morlan's
      theorem }
  WtFactor :=
    SumOfWeights ( LinkNode , ErrorExit , Arc , Node )   + 1;
  Arc . Weight [ PenArc ]   :=
    Arc . Weight [ PenArc ]   * WtFactor;
  Arc . Head [ PenArc ]   := LinkNode;

  PurgeNodes ( Arc, Node );

END; { - PROCEDURE FormTheJoint }


{ ----- PROCEDURE Triangulate -----
  - Input:
```

```
    -- Parameters:
       --- ReplArc:  the number of the arc to delete by
           triangulation
       --- HoldArc:  the number of the arc to make a circuit
           with
       --- Arc:  arc information
    -- Variables:  none
    -- I/O:  none
  - Action:  eliminates ReplArc by creating a circuit through
    HoldArc and moving ReplArc to the third side of the
    triangle
  - Output:
    -- Parameters:
       --- ErrorExit:  a BOOLEAN variable indicating the
           routine could not triangulate
       --- Arc:  updated arc information
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  20 Jan 92 }

PROCEDURE Triangulate
( ReplArc,
  HoldArc: TArcNum;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc );

BEGIN { - PROCEDURE Triangulate }

  ErrorExit := FALSE;   { - the default setting }

  WITH Arc DO
    BEGIN

      IF ( Head [ ReplArc ] = Head [ HoldArc ] ) OR
         ( Tail [ ReplArc ] = Tail [ HoldArc ] )
      THEN
        ReverseArc ( ReplArc , Arc );

      IF ( Head [ ReplArc ] = Tail [ HoldArc ] ) AND
         ( Tail [ ReplArc ] <> Head [ HoldArc ] )
      THEN
        BEGIN { - ReplArc points to tail of HoldArc }

          Head [ ReplArc ] := Head [ HoldArc ];
          { Tail [ ReplArc ] doesn't change }
          Weight [ ReplArc ] :=
            Weight [ ReplArc ] * Weight [ HoldArc ];

        END { - ReplArc points to tail of HoldArc }
      ELSE { - ReplArc doesn't point to tail of HoldArc }
        BEGIN

          IF ( Tail [ Rep   rc ] = Head [ HoldArc ] ) AND
             ( Head [ Rep   rc ] <> Tail [ HoldArc ] )
          THEN
```

```
            BEGIN { - HoldArc points to tail of ReplArc }

              Tail [ ReplArc ] := Head [ ReplArc ];
              Head [ ReplArc ] := Tail [ HoldArc ];
              Weight [ ReplArc ] :=  1.0 /
                ( Weight [ ReplArc ] * Weight [ HoldArc ] );

            END { - HoldArc points to tail of ReplArc }
          ELSE { - neither arc points to either end of
                   the other }

            ErrorExit := TRUE; { - can't triangulate }

        END; { - ReplArc doesn't point to tail of HoldArc }
    END; { - WITH Arc }
END; { - PROCEDURE Triangulate }


{ ----- PROCEDURE PurgeNodes -----
  - Input:
    -- Parameters:
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:  looks through full list of nodes for nodes not
    connected to the net; deletes them
  - Output:
    -- Parameters:
       --- Node:  updated node information
    -- Variables:  none
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:
    -- ScratchNode
  - Last modified:  20 Jan 92 }


PROCEDURE PurgeNodes
( VAR Arc: TArc;
  VAR Node: TNode );

VAR
  LocNumNodes: INTEGER;
  LocArcNum: TArcNum;
  OkayToDeleteNode: BOOLEAN;

BEGIN { - PROCEDURE PurgeNodes }

  { - search through all the nodes }
  LocNumNodes :=   Node . NNodes;
  REPEAT { UNTIL ( LocNumNodes = 0 ) }
    OkayToDeleteNode := TRUE;
    LocArcNum := 1;
    REPEAT { UNTIL ( LocArcNum >   Arc . NArcs   ) OR
                   ( OkayToDeleteNode = FALSE ) }
```

```
      IF (   Arc . NArcs   > 0 ) AND
         (   (   Arc . Head [ LocArcNum ]   = LocNumNodes ) OR
             (   Arc . Tail [ LocArcNum ]   = LocNumNodes )   )
      THEN
        OkayToDeleteNode := FALSE;
      INC ( LocArcNum );
    UNTIL ( LocArcNum >   Arc . NArcs   ) OR
          ( OkayToDeleteNode = FALSE );

    { - delete if okay }
    IF ( OkayToDeleteNode )
    THEN
      ScratchNode ( LocNumNodes , Arc , Node );

    DEC ( LocNumNodes );
  UNTIL ( LocNumNodes = 0 );

END; { - PROCEDURE PurgeNodes }


{ ----- PROCEDURE ReverseArc -----
  - Input:
    -- Parameters:
       --- ArcToRev:  the number of the arc to reverse
       --- Arc:  arc information
    -- Variables:  none
    -- I/O:  none
  - Action:  reverses arc ArcToRev
  - Output:
    -- Parameters:
       --- Arc:  updated arc information
    -- Variables:  none
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:  none
  - Last modified:  20 Jan 92 }

PROCEDURE ReverseArc
( ArcToRev : TArcNum;
  VAR Arc: TArc );

VAR
  Temp: TNodeNum;

BEGIN { - PROCEDURE ReverseArc }
  IF ( 0 < ArcToRev ) AND ( ArcToRev <=   Arc . NArcs   )
  THEN
    WITH Arc DO
      BEGIN
        Temp := Head [ ArcToRev ];
        Head [ ArcToRev ] := Tail [ ArcToRev ];
        Tail [ ArcToRev ] := Temp;
        Weight [ ArcToRev ] :=   1.0 / Weight [ ArcToRev ];
      END; { - WITH Arc }
END; { - PROCEDURE ReverseArc }
```

```
{ ----- PROCEDURE ScratchNode -----
  - Input:
    -- Parameters:
       --- NodeToScr:  the number of the node to scratch
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:  deletes a node from a net
  - Output:
    -- Parameters:
       --- Arc:  updated arc information
       --- Node:  updated node information
    -- Variables:  none
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:  none
  - Last modified:  20 Jan 92 }


PROCEDURE ScratchNode
( NodeToScr: TNodeNum;
  VAR Arc: TArc;
  VAR Node: TNode );

VAR
  LocArc: TArcNum;
  LocNode: TNodeNum;

BEGIN { - PROCEDURE ScratchNode }

  WITH Node DO
    BEGIN

      { - decrement the number of nodes }
      DEC (   Node . NNodes   );

      { - move higher-numbered nodes forward }
      IF ( NNodes > 0 ) THEN
        FOR LocNode := NodeToScr TO NNodes DO
          BEGIN
            NodeLabel [ LocNode ] :=
              NodeLabel [ LocNode + 1 ];
            EmbedArc  [ LocNode ] :=
              EmbedArc  [ LocNode + 1 ];
          END; { - FOR LocNode }

    END; { - WITH Node }

  IF (   Arc . NArcs   > 0 )
  THEN
    WITH Arc DO

      {- decrement appropriate node pointers in arc }
      FOR LocArc := 1 TO NArcs DO
```

```
        BEGIN

          IF ( Tail [ LocArc ] > NodeToScr )
          THEN
            DEC ( Tail [ LocArc ] );

          IF ( Head [ LocArc ] > NodeToScr )
          THEN
            DEC ( Head [ LocArc ] );

        END; { - FOR LocArc }

END; { - PROCEDURE ScratchNode }



{ ----- FUNCTION SumOfWeights -----
  - Input:
    -- Parameters:
       --- NodeToSumAt:  the number of the node at which to
           find the sum of all related arcs.
           ---- This routine will order DirectArcsToNode at
                this node, presuming that the action will
                affect an embedded net rather than the whole
                net.
       --- Arc:  arc information
       --- Node:  node information
    -- Variables:  none
    -- I/O:  none
  - Action:  computes and reports the sum of the weights
    of all vectors with NodeAtHead at the head
  - Output:
    -- Parameters:
       --- SumOfWeights:  the sum
    -- Variables:  none
    -- I/O:  none
  - Screen mode:  either text or graphics
  - External subroutines used:  none
  - Last modified:  31 Jan 92 }

FUNCTION SumOfWeights
( NodeToSumAt: TNodeNum;
  VAR ErrorExit: BOOLEAN;
  VAR Arc: TArc;
  VAR Node: TNode ): REAL;

VAR
  Wt: REAL;
  LocArc: TArcNum;
  NArcToPoint: INTEGER;

BEGIN { - FUNCTION SumOfWeights }

  SumOfWeights := 0; { - in case of ErrorExit }

  DirectArcsToNode ( NodeToSumAt , NArcToPoint , ErrorExit ,
```

```
      Arc , Node );
    IF ( ErrorExit )
    THEN
      EXIT; { - from FUNCTION SumOfWeights }

    Wt := 0;
    IF (   Arc . NArcs   > 0 )
    THEN
      FOR LocArc := 1 TO   Arc . NArcs   DO
        IF (   Arc . Head [ LocArc ]   = NodeToSumAt )
        THEN
          Wt := Wt +   Arc . Weight [ LocArc ];

    SumOfWeights := Wt;

  END; { - FUNCTION SumOfWeights }


END.
```

C.2 Source Code for the PROJPRST Unit

```
{ - VERSION:  Final }

UNIT ProjPrSt;

INTERFACE { ------------------------------------------------}

USES
  CRT,
  GRAPH,
  PrnThrm;

{ - This UNIT is part of the deliverable
    code for Maj Garry Flemings' thesis at AFIT. }

  { ------------------------------------------------------------
        START OF CONSTANTS USED TO SIZE DATA STRUCTURES
    ------------------------------------------------------------ }
  CONST

    MaxNumSensors = 10;
    MaxNumRptPerSensor = 256;
    MaxNumIDForTarget = 80;
    MaxNumActions = 80;

    MinNumRptPerSensor = 2;

    LowIdent = 0;
    LowAct = 0;
    LowSensor = 0;
    LowSensorRead = 0;
    LowUtilNum = 0;

  { ------------------------------------------------------------
```

```
        END OF CONSTANTS USED TO SIZE DATA STRUCTURES
    ------------------------------------------------------------ }

  TYPE
    TFullString = STRING [ 255 ];
    TStringIndex = 0 .. 255;

    RangeAct = LowAct .. (   MaxNumActions - 1   );
    RangeID = LowIdent  .. (   MaxNumIDForTarget - 1   );

    RangeSensorNums = LowSensor .. ( MaxNumSensors - 1 );
    RangeReadings = LowSensorRead .. ( MaxNumRptPerSensor - 1 );
    TROWVector = ARRAY [ RangeSensorNums ] OF RangeReadings;

    RangeUtils = LowUtilNum ..
      (   MaxNumActions * MaxNumIDForTarget   - 1 );
    TLocalUtils = ARRAY [ RangeUtils ] OF REAL;

    TDistributionOfID = ARRAY [ RangeID ] OF REAL;

  PROCEDURE PressAnyKey;

  PROCEDURE ProcessStarter
  ( VAR PIDFileTitle,
        PACFileTitle,
        ESPFileTitle,
        ROWFileTitle,
        PRRFileTitle,
        USFFileTitle: TFullString;
    VAR Out: TEXT;
    VAR ErrorExit: BOOLEAN;
    VAR HighIdent: RangeID;
    VAR HighAct: RangeAct;
    VAR ROWVector: TROWVector;
    VAR LocalUtils: TLocalUtils;
    VAR Arc: TArc;
    VAR Node: TNode );


IMPLEMENTATION { ----------------------------------------------}

  { ------------------------------------------------------------
        START OF DECLARATIONS AND PRIMITIVE ROUTINES
    ------------------------------------------------------------ }

  { definition, primitive routine:  a routine allowed full
    scope of the code.  Typically, a primitive routine does a
    small task used in unrelated parts of the code.

    In general, the programming style used here allows variables
    and routines to have no more scope than required for
    function.  Primitive routines are an exception to that rule.
    TYPE declarations may have full scope of the code and are
    placed for readability. }

  CONST
```

```
  MinSensorRptNum =   MinNumRptPerSensor - 1;
  MaxSensorRptNum =   MaxNumRptPerSensor - 1;
  MaxFieldNum = 65535;
  LowFieldNum = 0;

TYPE

  RangeSensorRptNum =  MinSensorRptNum .. MaxSensorRptNum;
  TSensor = ARRAY [ RangeSensorNums ] OF RangeSensorRptNum;
  TLocalESP =
    RECORD
      HighSensor: RangeSensorNums;
      HighRptNum: TSensor;
    END; { - TLocalESP RECORD }

  FileOfChar = FILE OF CHAR;

  RangeField = LowFieldNum .. MaxFieldNum;


{ ----- PROCEDURE PressAnyKey -----
  - Input:
    -- Parameters:  none
    -- Variables:  none
    -- I/O:  user hits any key, indicating readiness to
       continue
  - Action:  asks user to indicate readiness to continue
  - Output:
    -- Parameters:  none
    -- Variables:  none
    -- I/O:  screen prompts only
  - External subroutines used:  none
  - Last modified:  19 Nov 91 }

PROCEDURE PressAnyKey;

VAR
  RespChar: CHAR;

BEGIN { - PROCEDURE PressAnyKey }
  WRITELN; { - NOT to standard output; might be a disk file }
  WRITE ( 'Press any key to continue . . .' );
  RespChar := READKEY;
  WRITELN;
END; { - PROCEDURE PressAnyKey }



{ ------------------------------------------------------------
        END OF DECLARATIONS AND PRIMITIVE PROCEDURES
  ------------------------------------------------------------ }


{ ----- PROCEDURE ProcessStarter -----
  - Input:
    -- Parameters:
```

--- PIDFileTitle, PACFileTitle, ESPFileTitle, ROWFileTitle, PRRFileTitle, USFFileTitle: the MS-DOS file titles for six of the input files. In each case, the data passed must be associated with a file with the MS-DOS file extension of the same three letters as the start of the field (PID, PAC, ESP, etc.). The extension is allowed in the data, but not required.
--- Out: the TEXT variable for output
---- Within ProcessStarter, "Out" is treated as a GLOBAL VARIABLE rather than being passed thru parameter lists.
-- Variables: none
-- I/O:
--- Out: the TEXT variable for output. ProjModU uses "ASSIGN (Out , '')" to send the output to standard output and support redirection from the command line.
--- files: for each of the six input parameters listed above, this routine looks for a corresponding file. Additionally, for each line in the file with PIDFileTitle, the routine finds an associated file with MS-DOS file extensions of ".IDS" and ".IDU" for the remaining two input files
- Action: reads the input files; reformats the data into the internal representation used by the remainder of the program
- Output:
-- Parameters:
--- PIDFileTitle, PACFileTitle, ESPFileTitle, ROWFileTitle, PRRFileTitle, USFFileTitle: file titles, with appropriate extensions appended, if necessary
--- Out: the TEXT variable for output
--- ErrorExit: TRUE iff the code found an error from which it cannot recover
--- HighIdent: the highest identity number in the ".PID" file
--- HighAct: the high action number
--- ROWVector: a vector representing the reports coming from the sensors. The first position reports the first sensor, etc. Each report indicates the position on the sensor's discrete list for the report.
--- LocalUtils: a matrix flattened into a singly-dimensioned array. If two dimensional, the dimensions would be all possible identities and all possible actions. The values reflect the utility associated with each combination. This array is organized with all identities for one action, then all identities for the next action, etc.
--- Arc, Node: structures from the PRNTHRM UNIT for holding a probability ratio net. This net has two level of hierarchy. At the top level of

```
            hierarchy are as many nodes as there are
            identities.  Within each of those nodes is a
            probability ratio net indicating the relative
            probability of the given sensor reports to the
            sum of all other possible reports.
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:  none
    - Last modified:  22 Feb 92 }

PROCEDURE ProcessStarter
( VAR PIDFileTitle,
      PACFileTitle,
      ESPFileTitle,
      ROWFileTitle,
      PRRFileTitle,
      USFFileTitle: TFullString;
  VAR Out: TEXT;
  VAR ErrorExit: BOOLEAN;
  VAR HighIdent: RangeID;
  VAR HighAct: RangeAct;
  VAR ROWVector: TROWVector;
  VAR LocalUtils: TLocalUtils;
  VAR Arc: TArc;
  VAR Node: TNode );

  { ------------------------------------------------------
              START OF ProcessStarter PRIMITIVES
    ------------------------------------------------------ }



  { ----- PROCEDURE MsgBadData -----
    - Input:
      -- Parameters:
         --- FileTitle:  the name of the input file containing
             bad data
         --- OutStr:  the bad data field
         --- ProbChar:  the position in the data field of the
             offending character
      -- Variables:  none
      -- I/O:  none
    - Action:  puts error messages on the screen
    - Output:
      -- Parameters:  none
      -- Variables:  none
      -- I/O:  screen messages
    - External subroutines used:
      -- PressAnyKey
    - Last modified:  30 Jan 92 }

  PROCEDURE MsgBadData
  ( VAR FileTitle,
        OutStr: TFullString;
      { - VAR to save memory; used only for input }
    VAR ProbChar: INTEGER );
```

```
    { - VAR to save memory; used only for input }

BEGIN { - PROCEDURE MsgBadData }
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out , 'Error termination' );
  WRITELN ( Out );
  WRITELN ( Out , 'In reading ' , FileTitle ,
    ', here is an unreadable field:' );
  WRITELN ( Out , OutStr );
  WRITELN ( Out , '^' : ProbChar ,
    '   This character is unreadable.' );
  WRITELN ( Out );
  WRITELN ( Out , 'The program cannot recover.  ' );
  PressAnyKey;
END; { - PROCEDURE MsgBadData }


{ ----- PROCEDURE MsgExtraInput -----
  - Input:
    -- Parameters:
       --- FileTitle:  the title of the file with extra
           input fields
       --- HighIdent:  the high numbered identity in the
           problem
    -- Variables:  none
    -- I/O:  none
  - Action:  prints error messages to screen
  - Output:
    -- Parameters:  none
    -- Variables:  none
    -- I/O:  screen messages
  - External subroutines used:  none
  - Last modified:  30 Jan 92 }

PROCEDURE MsgExtraInput
( VAR FileTitle: TFullString;
    { - VAR to save memory; used only for input }
  NumNeeded: RangeID );

BEGIN { - PROCEDURE MsgExtraInput }
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out , 'Warning about ' , FileTitle , ':' );
  WRITELN ( Out , 'There should be ' , NumNeeded ,
    ' fields and there are more.' );
  WRITELN ( Out , 'Extra inputs are ignored.' );
END; { - PROCEDURE MsgExtraInput }


{ ----- PROCEDURE MsgNotEnoughData -----
  - Input:
    -- Parameters:
```

```
        --- FileTitle:  the name of the file which has too
            little data
        --- FieldNum:  the number of fields in the file with
            too little data
      -- Variables:  none
      -- I/O:  none
    - Action:  puts error messages on the screen
    - Output:
      -- Parameters:  none
      -- Variables:  none
      -- I/O:  screen messages
    - External subroutines used:
      -- PressAnyKey
    - Last modified:  30 Jan 92 }


PROCEDURE MsgNotEnoughData
( VAR FileTitle: TFullString;
    { - VAR to save memory; used only for input }
  VAR FieldNum: RangeField );
    { - VAR to save memory; used only for input }

BEGIN { - PROCEDURE MsgNotEnoughData }
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out , 'Error Termination' );
  WRITELN ( Out );
  WRITELN ( Out , 'In ' , FileTitle , ' there are only ' ,
    FieldNum , ' data fields.' );
  WRITELN ( Out , 'The file needs to have more fields.' );
  PressAnyKey;
END; { - PROCEDURE MsgNotEnoughData }



{ ----- PROCEDURE MsgNotEnoughValues -----
    - Input:
      -- Parameters:  same meanings as in parent
      -- Variables:  none
      -- I/O:  none
    - Action:  writes error messages to the screen
    - Output:
      -- Parameters:  none
      -- Variables:  none
      -- I/O:  screen messages
    - External subroutines used:
      -- PressAnyKey
    - Last modified:  28 Jan 92 }

PROCEDURE MsgNotEnoughValues
( VAR FileTitle: TFullString;
    { - VAR to save memory; used only for input }
  NumNeeded,
  NumThere: RangeID );
```

```
BEGIN { - PROCEDURE MsgNotEnoughValues }
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out );
  WRITELN ( Out , 'Error Termination' );
  WRITELN ( Out );
  WRITELN ( Out , 'The file ' , FileTitle ,
    ' has ' , NumThere , ' fields.' );
  WRITELN ( Out , 'There should be ' , NumNeeded , '.' );
  PressAnyKey;
END; { - PROCEDURE MsgNotEnoughValues }


{ ----- PROCEDURE Parser -----
  - Input:
    -- Parameters:
       --- InFile:  the file variable from which to read
    -- Variables:  none
    -- I/O:
       --- disk:  file associated with InFile
  - Action:  parses the incoming string of information from
    InFile into the units needed by the remainder of the
    program
    -- ignores everything on a line after %
    -- properly recognizes delimeters
  - Output:
    -- Parameters:
       --- OutStr:  a parsed input from InFile
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  26 Jan 92 }

PROCEDURE Parser
( VAR InFile: FileOfChar;
  VAR OutStr: TFullString );

CONST
  CR =   CHR ( 13 );
  Space = ' ';          { - CHR (  32 ) }
  Exclaim = '!';        { - CHR (  33 ) }
  Dollar = '$';         { - CHR (  36 ) }
  PerCent = '%';        { - CHR (  37 ) }
  Amper = '&';          { - CHR (  38 ) }
  Tilde = '~';          { - CHR ( 126 ) }

VAR
  InChar: CHAR;
  Done: BOOLEAN;

BEGIN { - PROCEDURE Parser }

  { - initialize }
  OutStr := '';
```

```
  REPEAT { - UNTIL ( Done ) }

    Done :=   EOF ( InFile );
    IF ( NOT Done )
    THEN
      BEGIN

        READ ( InFile , InChar );

        CASE InChar OF

          CR,
          Space:
            Done := (   LENGTH ( OutStr )   > 0 );

          Exclaim .. Dollar:
            OutStr :=   CONCAT ( OutStr , InChar );

          PerCent:  { - ignore everything to the CR }
            BEGIN
              REPEAT { - UNTIL ( Done ) OR
                                 ( InChar = CR ) }
                Done :=   EOF ( InFile );
                IF ( NOT Done )
                THEN
                  READ ( InFile , InChar );
              UNTIL ( Done ) OR ( InChar = CR );
              Done := ( Done ) OR
                      (   LENGTH ( OutStr )   > 0 );

            END; { - CASE InChar ; Comment }

          Amper .. Tilde:
            OutStr :=   CONCAT ( OutStr , InChar );

          { - no ELSE clause, because the routine should
              ignore every character not listed }
        END; { - CASE InChar }

      END; { - IF ( NOT Done ) }

  UNTIL ( Done );

END; { - PROCEDURE Parser }



{ ------------------------------------------------------
            START OF ProcessStarter PRIMITIVES
  ------------------------------------------------------ }


{ ----- PROCEDURE ESPReader -----
  - Input:
    -- Parameters:
       --- ESPFileTitle: the title of the file from which
```

```
            to read the description of the sensor suite
            (expected sensor performance)
      -- Variables:
         --- LowSensor
      -- I/O:
         --- file:  associated with ESPFileTitle
    - Action:  reads and interprets ESPFileTitle, putting
      the same information in LocalESP for the remainder of
      the code to use
    - Output:
      -- Parameters:
         --- ErrorExit:  TRUE iff the code found an error from
             which it cannot recover
         --- ESPFileTitle:  title of file, with ".ESP"
             appended if necessary
         --- LocalESP:  the description of the sensor suite
             performance
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:
      -- Parser
      -- PressAnyKey
    - Last modified:  21 Feb 92 }

PROCEDURE ESPReader
( VAR ErrorExit: BOOLEAN;
  VAR ESPFileTitle: TFullString;
  VAR LocalESP: TLocalESP );

CONST
  NoProblem = 0; { - return from VAL }

  { ----- PROCEDURE CheckMaxFieldNum -----
    - Input:
      -- Parameters:
         --- LocalESP:  specifications of the sensor suite
      -- Variables:  none
      -- I/O:  none
    - Action:  checks compliance with the limit for max
      number of distinct ROW reports; terminates if in
      violation
    - Output:
      -- Parameters:
         --- ErrorExit:  TRUE iff the sensor suite has
             too many reports
      -- Variables:  none
      -- I/O:  screen messages, if there are too many
         reports
    - External subroutines used:
      -- PressAnyKey
    - Last modified:  21 Feb 92 }

  PROCEDURE CheckMaxFieldNum
  ( VAR LocalESP: TLocalESP;
    { - VAR to save memory; used only for input }
    VAR ErrorExit: BOOLEAN );
```

```
VAR
  NeededFieldSize: REAL;
  SensorNum: RangeSensorNums;

BEGIN { - PROCEDURE CheckMaxFieldNum }
  NeededFieldSize := 1;
  FOR SensorNum := LowSensor TO   LocalESP . HighSensor
  DO
    NeededFieldSize := NeededFieldSize *
      (   LocalESP . HighRptNum [ SensorNum ]   + 1 );
  IF ( NeededFieldSize > MaxFieldNum )
  THEN
    BEGIN
      WRITELN ( Out );
      WRITELN ( Out );
      WRITELN ( Out );
      WRITELN ( Out , 'Error Termination' );
      WRITELN ( Out );
      WRITELN ( Out ,
        'Your sensor suite can generate up to ' ,
        NeededFieldSize : 2 : 0 , ' reports.' );
      WRITELN ( Out , '(Confirm that by multiplying the ' ,
        'number of reports for each sensor.) ' );
      WRITELN ( Out , 'The maximum is ' , MaxFieldNum ,
        '.' );
      PressAnyKey;

      ErrorExit := TRUE;

    END; { - IF ( NeededFieldSize > MaxFieldNum ) }
END; { - PROCEDURE CheckMaxFieldNum }



{ ----- PROCEDURE MsgTooBig -----
  - Input:
    -- Parameters:  same meaning as in parent
    -- Variables:  none
    -- I/O:  none
  - Action:  puts error messages on the screen
  - Output:
    -- Parameters:  none
    -- Variables:  none
    -- I/O:  screen messages
  - External subroutines used:
    -- PressAnyKey
  - Last modified:  30 Jan 92 }

PROCEDURE MsgTooBig
( ESPFileTitle,
  ESPOutStr: TFullString );

BEGIN { - PROCEDURE MsgTooBig }
  WRITELN ( Out );
```

```
    WRITELN ( Out );
    WRITELN ( Out );
    WRITELN ( Out , 'Error termination' );
    WRITELN ( Out );
    WRITELN ( Out , 'In reading ' , ESPFileTitle ,
      ' the program found this field' );
    WRITELN ( Out , ESPOutStr );
    WRITELN ( Out , 'No field can exceed ' ,
      MaxNumRptPerSensor , '.' );
    PressAnyKey;
  END; { - PROCEDURE MsgTooBig }


CONST
  NotThere = 0; { - for POS }

VAR
  ESPFile: FileOfChar;
  ESPOutStr: TFullString;
  SensorNum: RangeSensorNums;
  ProbChar,
  NumDistinctRpt: INTEGER;

BEGIN { - PROCEDURE ESPReader }

  { - assign the file variable a filename }
  IF (   POS ( '.' , ESPFileTitle )   = NotThere )
  THEN
    ESPFileTitle :=   CONCAT ( ESPFileTitle , '.ESP' );
  ASSIGN ( ESPFile , ESPFileTitle );
  RESET ( ESPFile );

  { - initialize }
  SensorNum := LowSensor;

  WHILE ( NOT   EOF ( ESPFile )   )
  DO
    BEGIN
      Parser ( ESPFile , ESPOutStr );
      IF (   LENGTH ( ESPOutStr )   > 0 )
      THEN
        BEGIN
          VAL ( ESPOutStr , NumDistinctRpt , ProbChar );
          IF ( ProbChar <> NoProblem )
          THEN
            BEGIN
              MsgBadData ( ESPFileTitle , ESPOutStr ,
                ProbChar );
              ErrorExit := TRUE;
              EXIT; { - from ESPReader }
            END; { - ProbChar <> NoProblem }

          IF ( NumDistinctRpt > MaxNumRptPerSensor )
          THEN
            BEGIN
              MsgTooBig ( ESPFileTitle , ESPOutStr );
```

```
            ErrorExit := TRUE;
            EXIT; { - from ESPReader }
          END; { - IF ( NumDistinctRpt >
                        NumRptPerSensor ) }

        { - The input is the number of distinct reports;
            the program stores the number of the highest
            report.  Adapt. }
        LocalESP . HighRptNum [ SensorNum ]   :=
          NumDistinctRpt - 1;
        INC ( SensorNum );
      END; { - IF (   LENGTH ( ESPOutStr )   > 0 ) }
    END; { - more ESP info }

  CLOSE ( ESPFile );

  LocalESP . HighSensor   := SensorNum - 1;

  CheckMaxFieldNum ( LocalESP , ErrorExit );

END; { - PROCEDURE ESPReader }



{ ----- PROCEDURE IDCounter -----
  - Input:
    -- Parameters:
       --- PIDFileTitle:  title of file in which to count
           the number of identities
    -- Variables:
       --- LowIdent
    -- I/O:
       --- disk:  file associated with PIDFileTitle
  - Action:  counts the number of fields reported in
    PIDFileTitle; reports one less as HighIdent
  - Output:
    -- Parameters:
       --- PIDFileTitle:  title of file, with ".PID"
           appended, if necessary
       --- HighIdent:  the number of identities reported
           in PIDFileTitle
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  25 Jan 92 }

PROCEDURE IDCounter
( VAR PIDFileTitle: TFullString;
  VAR HighIdent: RangeID );

CONST
  NotThere = 0; { - for POS }

VAR
  PIDFile: FileOfChar;
  PIDOutStr: TFullString;
```

```
BEGIN { - PROCEDURE IDCounter }
  IF (   POS ( '.' , PIDFileTitle )   = NotThere )
  THEN
    PIDFileTitle :=   CONCAT ( PIDFileTitle , '.PID' );
  ASSIGN ( PIDFile , PIDFileTitle );
  RESET ( PIDFile );
  HighIdent := LowIdent;

  WHILE ( NOT   EOF ( PIDFile )   )
  DO
    BEGIN
      Parser ( PIDFile , PIDOutStr );
      IF (   LENGTH ( PIDOutStr )   > 0 )
      THEN
        INC ( HighIdent );
    END; { - more PID }

  { - HighIdent currently holds the number of identities.
      They're numbered 0 .. ( HighIdent - 1 ).  Return
      the high index. }
  DEC ( HighIdent );

  CLOSE ( PIDFile );

END; { - PROCEDURE IDCounter }


{ ----- PROCEDURE PACReader -----
  - Input:
    -- Parameters:
       --- PACFileTitle:  title of file holding the list
           of actions to consider
    -- Variables:
       --- LowAct
    -- I/O:
       --- disk:  file associated with PACFileTitle
  - Action:  counts the number of lines in PACFileTitle;
    reports one less than that as HighAct
  - Output:
    -- Parameters:
       --- PACFileTitle:  title of file holding te list
           of actions to consider, with ".PAC" appended
           if necessary
       --- HighAct:  the high action number
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  26 Jan 92 }


PROCEDURE PACReader
( VAR PACFileTitle: TFullString;
  VAR HighAct: RangeAct );

CONST
```

```
  NotThere = 0; { - for POS }

VAR
  PACFile: TEXT;
  PACInStr: TFullString;

BEGIN { - PROCEDURE PACReader }

  IF (   POS ( '.' , PACFileTitle )   = NotThere )
  THEN
    PACFileTitle :=   CONCAT ( PACFileTitle , '.PAC' );
  ASSIGN ( PACFile , PACFileTitle );
  RESET ( PACFile );

  HighAct := LowAct;
  WHILE ( NOT   EOF ( PACFile )   )
  DO
    BEGIN

      { - read the HighAct numbered action }
      READLN ( PACFile , PACInStr );

      { - delete leading blanks }
      WHILE (   LENGTH ( PACInStr )   > 0 ) AND
            ( PACInStr [ 1 ] = ' ' )
      DO
        DELETE ( PACInStr , 1 , 1 );

      { - prepare HighAct for the next record; it leads
          the READLN }
      IF (   LENGTH ( PACInStr )   > 0 )
      THEN
        INC ( HighAct );

    END; { - WHILE ( NOT   EOF ( PACFile )   ) }

  CLOSE ( PACFile );

  DEC ( HighAct ); { - take away the lead }

END; { - PROCEDURE PACReader }



{ ----- PROCEDURE PID_IDS_IDUReader -----
  - Input:
    -- Parameters:
       --- PIDFileTitle:  the name of the file from which
           to draw names of identities
       --- HighIdent:  the number index of the high-numbered
           identity
       --- ROWVector:  the report of the world for which
           the program is computing a recommendation
       --- LocalESP:  specification of sensor suite
           characteristics
       --- USFVector:  utility scaling factors
```

```
    -- Variables:
       --- LowIdent
    -- I/O:  none
  - Action:  for each identity in the PID file, reads the
    needed information in the associated IDU and IDS files
    and builds the associated data structures
  - Output:
    -- Parameters:
       --- LocalUtils:  the utility values for each
           identity-action pairing
       --- Arc:  the arcs of the PRN recording relationships
           between ROW and the identities
       --- Node:  the nodes of the same PRN
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- Parser
  - Last modified:  30 Jan 92 }

PROCEDURE PID_IDS_IDUReader
( VAR PIDFileTitle: TFullString;
    { - VAR to save memory; used only for input }
  HighIdent: RangeID;
  HighAct: RangeAct;
  VAR ROWVector: TROWVector;
    { - VAR to save memory; used only for input }
  VAR LocalESP: TLocalESP;
    { - VAR to save memory; used only for input }
  VAR USFVector: TDistributionOfID;
  VAR ErrorExit: BOOLEAN;
  VAR LocalUtils: TLocalUtils;
  VAR Arc: TArc;
  VAR Node: TNode );

  { ----- PROCEDURE BuildPRN -----
    - Input:
      -- Parameters:
         --- LocIdentNum:  the number of the identity the
             program is adding to the PRN
         --- ProbROW:  the probability of ROW given that
             the target of interest is represented by
             LocIdentNum
         --- Arc, Node:  the existing PRN
      -- Variables:  none
      -- I/O:  none
    - Action:  adds the information for LocIdentNum into
      the PRN
    - Output:
      -- Parameters:
         --- Arc, Node:  updated PRN
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:  none
    - Last modified:  30 Jan 92 }

  PROCEDURE BuildPRN
```

```
( LocIdentNum: RangeID;
  ProbROW: REAL;
  VAR Arc: TArc;
  VAR Node: TNode );

VAR
  Node1ToMake,
  Node2ToMake,
  LocNodeNum: TNodeNum;
  ArcToMake: TArcNum;

BEGIN { - PROCEDURE BuildPRN }

  ArcToMake :=   Arc . NArcs   + 1;
  Node1ToMake :=   Node . NNodes + 1;
  Node2ToMake :=   Node . NNodes + 2;

  { - find the node for LocIdentNum }
  LocNodeNum := LowNodeNum;
  WHILE (   Node . NodeLabel [ LocNodeNum ] [ 1 ]   <>
            CHR ( LocIdentNum )   )
  DO
    INC ( LocNodeNum );

  { - update parent node }
  Node . EmbedArc [ LocNodeNum ]   := ArcToMake;

  { - build new node 1 }
  Node . EmbedArc [ Node1ToMake ]   := NoEmbedArc;
  Node . NodeLabel [ Node1ToMake ]   :=
    Node . NodeLabel [ LocNodeNum ];

  { - build new node 2 }
  Node . EmbedArc [ Node2ToMake ]   := NoEmbedArc;
  Node . NodeLabel [ Node2ToMake ]   := '';

  { - build new arc }
  Arc . Head [ ArcToMake ]   := Node2ToMake;
  Arc . Tail [ ArcToMake ]   := Node1ToMake;
  Arc . Weight [ ArcToMake ]   :=
    ProbROW / (   1 - ProbROW   );
    { - ProbROW is the probability of the reported ROW.
      - (   1 - ProbROW   ) is the probability of all
        the other ROW's put together
      - the quotient is the probability ratio }

  { - update NArcs and NNodes }
  INC (   Arc . NArcs   );
  INC (   Node . NNodes   , 2 );

END; { - PROCEDURE BuildPRN }



{ ----- FUNCTION FindFieldAndGroup -----
  - Input:
```

```
  -- Parameters:
     --- ROWVector:  the report of the world for which
         the program is computing values
     --- LocalESP:  the characteristics of the sensor
         suite
  -- Variables:
     --- LowFieldNum
     --- LowSensor
  -- I/O:  none
- Action:  for IDS and IDU files, computes the relevant
  field number and action group size
  -- IDS files are ordered sets of values.  The first
     value is for the report of the the world (ROW)
     corresponding to the first-listed report of all
     sensors in the sensor suite.  The second value in
     the file is for the ROW corresponding to all the
     same readings, except it corresponds to the
     second-listed report of the last sensor.  Values
     for further reports of the last sensor follow in
     order.  A similar set of values follows next for
     the second-listed report of the next-to-last
     sensor.  The pattern continues until listing the
     value corresponding to the last-listed report of
     all sensors.
  -- IDU files are similar, but contain information,
     also, for each action being considered.  In
     particular, IDU files consist of multiple
     instances of the same information in a single IDS
     file.  The first instance is for the first listed
     action, then the second, and continuing until
     having listed the last action.
  -- This program needs only one value from each of
     these sets.  In particular, it needs only one
     value from the IDS file--the value corresponding
     to the reported ROW.  Additionally, it needs as
     many values from the IDU file as there are
     actions, in particular one value for each action.
     --- Because each set (the IDS file and information
         in the IDU file on each action) is of
         identical size and organization, the program
         needs an offset into the data set to find the
         needed data.
     --- Because the program needs to pick up the
         corresponding entries from each succeeding
         set, it needs the size of the data sets.
     --- Those values are the output:  FieldWanted and
         ActGroupSize.
- Output:
  -- Parameters:
     --- FieldWanted:  the offset within each action
         group holding the desired data
     --- ActGroupSize:  the number of fields in the
         data group holding information for one action
  -- Variables:  none
  -- I/O:  none
- External subroutines used:  none
```

```
  - Last modified:  29 Jan 92 }

PROCEDURE FindFieldAndGroup
( VAR ROWVector: TROWVector;
    { - VAR to save memory; used only for input }
  VAR LocalESP: TLocalESP;
    { - VAR to save memory; used only for input }
  VAR FieldWanted,
      ActGroupSize: RangeField );

VAR
  GroupWidth: RangeField;
  SensorNum: RangeSensorNums;

BEGIN { - PROCEDURE FindFieldAndGroup }
  FieldWanted := LowFieldNum;
  GroupWidth := 1;
  FOR SensorNum :=   LocalESP . HighSensor   DOWNTO
    LowSensor
  DO
    BEGIN
      INC ( FieldWanted ,
        GroupWidth * ROWVector [ SensorNum ] );
      GroupWidth := GroupWidth *
        (   LocalESP . HighRptNum [ SensorNum ]   + 1   );
    END; { - FOR SensorNum }
  ActGroupSize := GroupWidth;
END; { - PROCEDURE FindFieldAndGroup }


{ ----- PROCEDURE IDSReader -----
  - Input:
    -- Parameters:
       --- PIDOutStr:  the name of the identity to read
       --- FieldWanted:  the position within the IDS file
           of the desired ROW information
       --- GroupWidth:  the expected size of the IDS file,
           in fields
    -- Variables:
       --- LowFieldNum
    -- I/O:  none
  - Action:  finds the appropriate relative probability
    in the IDS file, computes the actual probability
  - Output:
    -- Parameters:
       --- ErrorExit:  TRUE iff the code found an error
           from which it cannot recover
       --- ThisProb:  the probability of the reported ROW
           given that the target of interest has the
           identity represented by PIDOutStr
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- Parser
  - Last modified:  21 Feb 92 }
```

```
PROCEDURE IDSReader
( VAR PIDOutStr: TFullString;
    { - VAR to save memory; used only for input }
  FieldWanted,
  GroupWidth: RangeField;
  VAR ErrorExit: BOOLEAN;
  VAR ThisProb: REAL );

  { ----- PROCEDURE MsgExtraData -----
    - Input:
      -- Parameters:  same meanings as parent
      -- Variables:  none
      -- I/O:  none
    - Action:  prints advisory messages on the screen
    - Output:
      -- Parameters:  none
      -- Variables:  none
      -- I/O:  screen messages
    - External subroutines used:  none
    - Last modified:  30 Jan 92 }

  PROCEDURE MsgExtraData
  ( IDSFileTitle: TFullString;
    HighFieldInFile: RangeField );

  BEGIN { - PROCEDURE MsgExtraData }
    WRITELN ( Out );
    WRITELN ( Out );
    WRITELN ( Out );
    WRITELN ( Out , 'Data Warning' );
    WRITELN ( Out );
    WRITELN ( Out , 'In ' , IDSFileTitle ,
      ', there should be ' ,   HighFieldInFile + 1    ,
      ' fields.' );
    WRITELN ( Out ,
      'There are more; the extras are ignored.' );
    WRITELN ( Out );
  END; { - PROCEDURE MsgExtraData }


CONST
  NoProblem = 0; { - VAL output }

VAR
  IDSOutStr,
  IDSFileTitle: TFullString;
  IDSFile: FileOfChar;
  ReadingDone: BOOLEAN;
  RelProb,
  SumOfAll: REAL;
  CharProblem: INTEGER;
  HighFieldInFile,
  FieldNum: RangeField;
```

```
BEGIN { - PROCEDURE IDSReader }

  IDSFileTitle :=   CONCAT ( PIDOutStr , '.IDS' );
  ASSIGN ( IDSFile , IDSFileTitle );
  RESET ( IDSFile );
  ReadingDone := FALSE;
  SumOfAll := 0;
  FieldNum := LowFieldNum;
  HighFieldInFile := GroupWidth - 1;

  REPEAT { - UNTIL ( ReadingDone ) }

    Parser ( IDSFile , IDSOutStr );

    IF (   LENGTH ( IDSOutStr )   = 0 )
    THEN
      BEGIN
        MsgNotEnoughData ( IDSFileTitle , FieldNum );
        ErrorExit := TRUE;
        EXIT; { - from IDSReader }
      END; { - IF (   LENGTH ( IDSOutStr )   = 0 ) }

    VAL ( IDSOutStr , RelProb , CharProblem );

    IF ( CharProblem <> NoProblem )
    THEN
      BEGIN
        MsgBadData ( IDSFileTitle , IDSOutStr ,
          CharProblem );
        ErrorExit := TRUE;
        EXIT; { - from IDSReader }
      END; { - IF ( CharProblem <> NoProblem ) }

    SumOfAll := SumOfAll + RelProb;

    IF ( FieldNum = FieldWanted )
    THEN
      ThisProb := RelProb;

    IF ( FieldNum = HighFieldInFile )
    THEN
      ReadingDone := TRUE
    ELSE
      INC ( FieldNum );

  UNTIL ( ReadingDone );

  Parser ( IDSFile , IDSOutStr );

  IF (   LENGTH ( IDSOutStr )   > 0 )
  THEN
    MsgExtraData ( IDSFileTitle , HighFieldInFile );

  CLOSE ( IDSFile );
```

```
  ThisProb := ThisProb / SumOfAll;

END; { - PROCEDURE IDSReader }


{ ----- PROCEDURE IDUReader -----
  - Input:
    -- Parameters:
       --- IDIndex:  the index of the identity being
           processed (0, 1, . . . )
       --- HighIdent:  the index of the last identity
           to process
       --- HighAct:  the high numbered action
       --- PIDOutStr:  the name of the file, without an
           extension, to read the IDU information from
       --- ThisUSFFactor:  the USF entry that applies to
           the identity being processed
       --- FieldWanted:  an index into the IDU file for
           the pieces of information needed
       --- ActGroupSize:  the separation in the IDU file
           of successive needed pieces of information
    -- Variables:
       --- LowFieldNum
    -- I/O:
       --- disk:  the file associated with PIDOutStr and
           ".IDU"
  - Action:  reads the necessary values from the
    applicable IDU file, modifies them with
    ThisUSFFactor, and returns the appropriate utilities
  - Output:
    -- Parameters:
       --- ErrorExit:  TRUE iff the code found an error
           from which it cannot recover
       --- LocalUtils:  values filled in corresponding to
           this combination of identity and actions
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- Parser
  - Last modified:  21 Feb 92 }


PROCEDURE IDUReader
( IDIndex,
  HighIdent: RangeID;
  HighAct: RangeAct;
  VAR PIDOutStr: TFullString;
    { - VAR to save memory; used only for input }
  ThisUSFFactor: REAL;
  FieldWanted,
  ActGroupSize: RangeField;
  VAR ErrorExit: BOOLEAN;
  VAR LocalUtils: TLocalUtils );

CONST
  NoProblem = 0; { - VAL output }
```

```
VAR
  IDUOutStr,
  IDUFileTitle: TFullString;
  IDUFile: FileOfChar;
  LastField,
  FieldNum: RangeField;
  RelUtil: REAL;
  NumID,
  CharProblem: INTEGER;
  ReadingDone: BOOLEAN;
  LUIndex: RangeUtils;

BEGIN { - PROCEDURE IDUReader }

  IDUFileTitle :=   CONCAT ( PIDOutStr , '.IDU' );
  ASSIGN ( IDUFile , IDUFileTitle );
  RESET ( IDUFile );

  LastField :=
    FieldWanted +   HighAct * ActGroupSize;
  ReadingDone := FALSE;
  FieldNum := LowFieldNum;
  Parser ( IDUFile , IDUOutStr );
  LUIndex := IDIndex;
  NumID := HighIdent + 1;

  REPEAT { - UNTIL ( ReadingDone ) }

    WHILE ( FieldNum < FieldWanted )
    DO
      BEGIN
        Parser ( IDUFile , IDUOutStr );
        INC ( FieldNum );
      END; { - WHILE ( FieldNum < FieldWanted ) }

    IF (   LENGTH ( IDUOutStr )   = 0 )
    THEN
      BEGIN
        MsgNotEnoughData ( IDUFileTitle , FieldNum );
        ErrorExit := TRUE;
        EXIT; { - from IDUReader }
      END; { - IF (   LENGTH ( IDUOutStr )   = 0 ) }

    VAL ( IDUOutStr , RelUtil , CharProblem );
    IF ( CharProblem <> NoProblem )
    THEN
      BEGIN
        MsgBadData ( IDUFileTitle , IDUOutStr ,
          CharProblem );
        ErrorExit := TRUE;
        EXIT; { - from IDUReader }
      END; { - IF ( CharProblem <> NoProblem ) }

    LocalUtils [ LUIndex ] := ThisUSFFactor * RelUtil;
    INC ( LUIndex , NumID );
```

```
        { - The relevant data is spread across LocalUtils
            at intervals of the number of identities. }

      IF ( FieldWanted = LastField )
      THEN
        ReadingDone := TRUE
      ELSE
        INC ( FieldWanted , ActGroupSize );

    UNTIL ( ReadingDone );

    CLOSE ( IDUFile );

  END; { - PROCEDURE IDUReader }

CONST
  ProbChar = 0; { - VAL output }


VAR
  PIDFile: FileOfChar;
  PIDOutStr: TFullString;
  FieldWanted,
  ActGroupSize: RangeField;
  LocIdentNum: RangeID;
  ProbROW: REAL;
  NodeLabel: TNodeStr; { - type from PRNTHRM UNIT }

BEGIN { - PROCEDURE PID_IDS_IDUReader }

  { - initializations }
  ASSIGN ( PIDFile , PIDFileTitle );
  RESET ( PIDFile );
  FindFieldAndGroup ( ROWVector , LocalESP ,
    FieldWanted , ActGroupSize );

  FOR LocIdentNum := LowIdent TO HighIdent
  DO
    BEGIN
      Parser ( PIDFile , PIDOutStr );

      IDUReader ( LocIdentNum , HighIdent , HighAct ,
        PIDOutStr , USFVector [ LocIdentNum ] ,
        FieldWanted , ActGroupSize ,
        ErrorExit , LocalUtils );
      IF ( ErrorExit ) THEN
        EXIT; { - from PID_IDS_IDUReader }

      IDSReader ( PIDOutStr , FieldWanted , ActGroupSize ,
        ErrorExit , ProbROW );
      IF ( ErrorExit ) THEN
        EXIT; { - from PID_IDS_IDUReader }

      BuildPRN ( LocIdentNum , ProbROW , Arc , Node );
    END; { - WHILE (   LENGTH ( PIDStr )   > 0 ) }
```

```
  CLOSE ( PIDFile );

END; { - PROCEDURE PID_IDS_IDUReader }



{ ----- PROCEDURE PRRReader -----
  - Input:
    -- Parameters:
       --- HighIdent:  the highest-numbered identity to
           consider
       --- PRRFileTitle:  the file title to read PRR
           information from
    -- Variables:
       --- LowIdent
    -- I/O:
       --- disk:  file associated with PRRFileTitle
  - Action:  reads the PRR file and computes the prior
    distribution
  - Output:
    -- Parameters:
       --- ErrorExit:  TRUE iff the code found an error from
           which it could not recover
       --- PRRFileTitle:  file title with ".PRR" appended,
           if necessary
       --- PRRVector:  a probability distribution on the
           possible identities
       --- Arc, Node:  initialized PRN
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- Parser
  - Last modified:  21 Feb 92 }

PROCEDURE PRRReader
( HighIdent: RangeID;
  VAR ErrorExit: BOOLEAN;
  VAR PRRFileTitle: TFullString;
    { - VAR to save memory; used only for input }
  VAR PRRVector: TDistributionOfID;
  VAR Arc: TArc;
  VAR Node: TNode );

CONST
  NoProblem = 0; { - output from VAL }

  { ----- PROCEDURE BuildPRN -----
    - Input:
      -- Parameters:
         --- PRRVector:  the report from the PRR file on
             the probability of the identities in the PID
             file
         --- HighIdent:  the index of the high-numbered
             identity in the problem
      -- Variables:
         --- LowIdent
```

```
    -- I/O:  none
  - Action:  creates the nodes and arcs in the first level
    of hierarchy for the PRN.  The structure it creates is
    a star--all nodes pointing to the first one.  The
    weights on the arcs are functions of PRRVector.
  - Output:
    -- Parameters:
       --- Arc, Node:  initialized PRN
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  30 Jan 92 }

PROCEDURE BuildPRN
( PRRVector: TDistributionOfID;
    { - not VAR:  changed for use only of this routine }
  HighIdent: RangeID;
  VAR Arc: TArc;
  VAR Node: TNode );

VAR
  IDNum: RangeID;
  NodeToMake: TNodeNum;
  ArcToMake: TArcNum;

BEGIN { - PROCEDURE BuildPRN }

  { - initialize Arc and Node to empty }
  Arc . NArcs   := 0;
  Node . NNodes   := 0;

  { - compute the probability ratios }
  FOR IDNum := (   LowIdent + 1 )   TO HighIdent
  DO
    PRRVector [ IDNum ] :=
      PRRVector [ IDNum ] / PRRVector [ LowIdent ];

  { - build the first hierarchy of nodes and arcs }
  FOR IDNum := LowIdent TO HighIdent
  DO
    BEGIN
      ArcToMake :=   Arc . NArcs   + 1;
      NodeToMake :=   Node . NNodes   + 1;

      Node . EmbedArc [ NodeToMake ]   := NoEmbedArc;
      Node . NodeLabel [ NodeToMake ]   := ' ';
        { - properly sets the string length to 1 }
      Node . NodeLabel [ NodeToMake ] [ 1 ]   :=
        CHR ( IDNum );
        { - changes the first character only }

      IF (   Node . NNodes   > 0 )
      THEN
        BEGIN
          Arc . Head [ ArcToMake ]   := 1;
          Arc . Tail [ ArcToMake ]   := NodeToMake;
```

```
            Arc . Weight [ ArcToMake ]   :=
              PRRVector [ IDNum ];
            INC (   Arc . NArcs   );
          END; { - IF (   Node . NNodes   > 0 ) }

        INC (   Node . NNodes );
      END; { - FOR IDNum }

  END; { - PROCEDURE BuildPRN }



CONST
  NotThere = 0; { - for POS }

VAR
  LocID: RangeID;
  PRROutStr: TFullString;
  SumOfInputs: REAL;
  PRRFile: FileOfChar;
  ErrorChar: INTEGER;

BEGIN { - PROCEDURE PRRReader }

  IF (   POS ( '.' , PRRFileTitle )   = NotThere )
  THEN
    PRRFileTitle :=   CONCAT ( PRRFileTitle , '.PRR' );
  ASSIGN ( PRRFile , PRRFileTitle );
  RESET ( PRRFile );

  SumOfInputs := 0;

  FOR LocID := LowIdent TO HighIdent
  DO
    BEGIN
      Parser ( PRRFile , PRROutStr );
      IF (   LENGTH ( PRROutStr )   = 0 ) THEN
      BEGIN
        MsgNotEnoughValues ( PRRFileTitle ,
          HighIdent + 1   , LocID );
        ErrorExit := TRUE;
        EXIT; { - from PRRReader }
      END; { - IF (   LENGTH ( PRROutStr )   = 0 ) }
      VAL ( PRROutStr , PRRVector [ LocID ] , ErrorChar );
      IF ( ErrorChar <> NoProblem )
      THEN
        BEGIN
          MsgBadData ( PRRFileTitle , PRROutStr ,
            ErrorChar );
          ErrorExit := TRUE;
          EXIT; { - from PRRReader }
        END; { - IF ( ErrorChar <> NoProblem ) }
      SumOfInputs := SumOfInputs + PRRVector [ LocID ];
    END; { - FOR LocID }

  { - compute an actual distribution
```

```
      from the relative values in the input }
    FOR LocID := LowIdent TO HighIdent
    DO
      PRRVector [ LocId ] :=
        PRRVector [ LocID ] / SumOfInputs;

    { - check for extra PRR values }
    Parser ( PRRFile , PRROutStr );
    IF (   LENGTH ( PRROutStr )   > 0 ) THEN
      MsgExtraInput ( PRRFileTitle ,   HighIdent + 1   );

    CLOSE ( PRRFile );

    BuildPRN ( PRRVector , HighIdent , Arc , Node );

END; { - PROCEDURE PRRReader }



{ ----- PROCEDURE ROWReader -----
    - Input:
      -- Parameters:
         --- LocalESP:  the definition of the acceptable
             sensor reports.  A RECORD of information.
         --- ROWFileTitle:  the title of the file in which
             to find the sensor reports
      -- Variables:
         --- LowSensor
      -- I/O:
         --- file:  associated with ROWFileTitle
    - Action:  reads and interprets ROWFileTitle; loads
      ROWVector with that information
    - Output:
      -- Parameters:
         --- ErrorExit:  TRUE iff the input file had too few
             values
         --- ROWFileTitle:  file title with ".ROW" appended,
             if necessary
         --- ROWVector:  the sensor reports.  The first
             position (numbered 0) has the report of the
             first sensor, etc. for as many sensors as are
             in the suite.  Each report is a zero-based
             integer representing one of the several possible
             reports from each sensor.
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:
      -- Parser
    - Last modified:  21 Feb 92 }

PROCEDURE ROWReader
( VAR LocalESP: TLocalESP;
    { - VAR to save memory; used only for input }
    VAR ErrorExit: BOOLEAN;
    VAR ROWFileTitle: TFullString;
    VAR ROWVector: TROWVector );
```

```
CONST
  NoProblem = 0; { - return from VAL }


  { ----- PROCEDURE MsgNotInRange -----
    - Input:
      -- Parameters:
         --- ROWFileTitle:  same as parent
         --- FieldNum:  the number of the offending field
             (1-based)
         --- InputWas:  the input of the offending field
         --- HighInputNum:  the high member of the
             acceptable range
      -- Variables:  none
      -- I/O:  none
    - Action:  puts error messages on the screen
    - Output:
      -- Parameters:  none
      -- Variables:  none
      -- I/O:  screen messages
    - External subroutines used:  none
    - Last modified:  29 Jan 92 }

  PROCEDURE MsgNotInRange
  ( ROWFileTitle: TFullString;
    FieldNum: RangeSensorNums;
    InputWas: INTEGER;
    HighInputNum: RangeSensorRptNum );

  BEGIN { - PROCEDURE MsgNotInRange }
    WRITELN ( Out );
    WRITELN ( Out );
    WRITELN ( Out );
    WRITELN ( Out , 'Data Error:  value out of range' );
    WRITELN ( Out );
    WRITELN ( Out , 'In file ' , ROWFileTitle , ', field ' ,
      FieldNum , ' is:  ' , InputWas );
    WRITELN ( Out , 'It should be in the range [1,' ,
      HighInputNum , '].' );
    WRITELN ( Out , 'Setting it to 1 and continuing.' );
  END; { - PROCEDURE MsgNotInRange }


CONST
  NotThere = 0; { - for POS }

VAR
  SensorNum: RangeSensorNums;
  ROWFile: FileOfChar;
  ROWOutStr: TFullString;
  SensorRead,
  ProbChar: INTEGER;

BEGIN { - PROCEDURE ROWReader }
```

```
  IF (   POS ( '.' , ROWFileTitle )   = NotThere )
  THEN
    ROWFileTitle :=   CONCAT ( ROWFileTitle , '.ROW' );
  ASSIGN ( ROWFile , ROWFileTitle );
  RESET ( ROWFile );

  FOR SensorNum := LowSensor TO   LocalESP . HighSensor
  DO
    BEGIN
      Parser ( ROWFile , ROWOutStr );
      IF (   LENGTH ( ROWOutStr )   = 0 )
      THEN
        BEGIN
          MsgNotEnoughValues ( ROWFileTitle ,
            LocalESP . HighSensor   + 1   , SensorNum );
        END; { - IF (   LENGTH ( PRROutStr )   = 0 ) }

      VAL ( ROWOutStr , SensorRead , ProbChar );
      DEC ( SensorRead );
        { - converts 1-based readings for user
            in ROW file to 0-based for program }
      IF ( ProbChar = NoProblem )
      THEN
        BEGIN
          IF ( SensorRead < LowSensor ) OR
             (   LocalESP . HighRptNum [ SensorNum ]   <
             SensorRead )
          THEN { - SensorRead is not in range }
          BEGIN
            MsgNotInRange ( ROWFileTitle , SensorNum ,
              SensorRead + 1   ,
              LocalESP . HighRptNum [ SensorNum ]   + 1 );
            SensorRead := 0;
          END { - IF ( SensorRead out of range ) }
        END { - IF ( ProbChar = NoProblem }
      ELSE { - ( ProbChar <> NoProblem ) }
        BEGIN
          MsgBadData ( ROWFileTitle , ROWOutStr , ProbChar );
          ErrorExit := TRUE;
          EXIT; { - from ROWReader }
        END; { - ( ProbChar <> NoProblem ) }
      ROWVector [ SensorNum ] := SensorRead;
    END; { - FOR SensorNum }

  { - check for extra ROW values }
  Parser ( ROWFile , ROWOutStr );
  IF (   LENGTH ( ROWOutStr )   > 0 ) THEN
    MsgExtraInput ( ROWFileTitle ,
      LocalESP . HighSensor   + 1 );

  CLOSE ( ROWFile );

END; { - PROCEDURE ROWReader }
```

```
{ ----- PROCEDURE USFReader -----
  - Input:
    -- Parameters:
       --- HighIdent:  the highest-numbered identity to
           be considered here
       --- USFFileTitle:  the name of the file from which
           to read utility scaling factors
    -- Variables:
       --- LowIdent
    -- I/O:
       --- disk:  file associated with USFFileTitle
  - Action:  reads the utility scaling factors from disk
    and puts them in USFVector
  - Output:
    -- Parameters:
       --- ErrorExit:  TRUE iff there weren't enough values
           in the file
       --- USFFileTitle:  file name, with ".USF" appended,
           if necessary
       --- USFVector:  the utility scaling factors
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- Parser
  - Last modified:  21 Feb 92 }

PROCEDURE USFReader
( HighIdent: RangeID;
  VAR ErrorExit: BOOLEAN;
  VAR USFFileTitle: TFullString;
    { - VAR to save memory; used only for input }
  VAR USFVector: TDistributionOfID );

CONST
  NoProblem = 0; { - output of VAL }
  NotThere = 0; { - for POS }

VAR
  USFFile: FileOfChar;
  LocID: RangeID;
  USFOutStr: TFullString;
  ErrorChar: INTEGER;

BEGIN { - PROCEDURE USFReader }

  IF (   POS ( '.' , USFFileTitle )   = NotThere )
  THEN
    USFFileTitle :=   CONCAT ( USFFileTitle , '.USF' );
  ASSIGN ( USFFile , USFFileTitle );
  RESET ( USFFile );

  FOR LocID := LowIdent TO HighIdent
  DO
    BEGIN
```

```
        Parser ( USFFile , USFOutStr );
        IF (   LENGTH ( USFOutStr )   = 0 ) THEN
        BEGIN
          MsgNotEnoughValues ( USFFileTitle ,
            HighIdent + 1    , LocID );
          ErrorExit := TRUE;
          EXIT; { - from USFReader }
        END; { - IF (   LENGTH ( PRROutStr )   = 0 ) }
        VAL ( USFOutStr , USFVector [ LocID ] , ErrorChar );
        IF ( ErrorChar <> NoProblem )
        THEN
          BEGIN
            MsgBadData ( USFFileTitle , USFOutStr ,
              ErrorChar );
            USFVector [ LocID ] := 0;
          END; { - IF ( ErrorChar <> NoProblem ) }
      END; { - FOR LocID }

    { - check for extra USF values }
    Parser ( USFFile , USFOutStr );
    IF (   LENGTH ( USFOutStr )   > 0 ) THEN
      MsgExtraInput ( USFFileTitle ,   HighIdent + 1   );

    CLOSE ( USFFile );

  END; { - PROCEDURE USFReader }


VAR
  LocalESP: TLocalESP;
  USFVector,
  PRRVector: TDistributionOfID;

BEGIN { - PROCEDURE ProcessStarter }

  { - file reading routines independent of all others }
  ESPReader ( ErrorExit , ESPFileTitle , LocalESP );
  IF ( ErrorExit ) THEN
    EXIT; { - from ProcessStarter }
  IDCounter ( PIDFileTitle , HighIdent );
  PACReader ( PACFileTitle , HighAct );

  { - file reading routines dependent on others }
  PRRReader ( HighIdent , ErrorExit , PRRFileTitle ,
    PRRVector , Arc , Node );
  IF ( ErrorExit ) THEN
    EXIT; { - from ProcessStarter }
  ROWReader ( LocalESP , ErrorExit , ROWFileTitle ,
    ROWVector );
  IF ( ErrorExit ) THEN
    EXIT; { - from ProcessStarter }
  USFReader ( HighIdent , ErrorExit , USFFileTitle ,
    USFVector );
  IF ( ErrorExit ) THEN
    EXIT; { - from ProcessStarter }
```

```
    PID_IDS_IDUReader ( PIDFileTitle , HighIdent , HighAct ,
      ROWVector , LocalESP , USFVector ,
      ErrorExit ,
      LocalUtils , Arc , Node );

  END; { - PROCEDURE ProcessStarter }


END.
```

C.3 Source Code for the PROJMODU Unit

```
{ - VERSION:  Final }

UNIT ProjModU;

INTERFACE { ------------------------------------------------}

USES
  CRT,
  GRAPH,
  PrnThrm,
  ProjPrSt;

{ - This UNIT is part of the deliverable
    code for Maj Garry Flemings' thesis at AFIT. }

  VAR
    Out: TEXT; { - text file for all
                   output; supports redirection }

  PROCEDURE ProjMod
  ( PIDFileTitle,
    PACFileTitle,
    ESPFileTitle,
    ROWFileTitle,
    PRRFileTitle,
    USFFileTitle: TFullString;
    VAR ErrorExit: BOOLEAN;
    VAR MostLikelyID,
        RecommendAction: INTEGER );


IMPLEMENTATION { ------------------------------------------------}

  { ----- PROCEDURE ProjMod -----
    - Input:
      -- Parameters:
         --- PIDFileTitle, PACFileTitle, ESPFileTitle,
             ROWFileTitle, PRRFileTitle, USFFileTitle:  the
             filenames of files defining the problem to work.
             Each file name has the same extension as the first
             three letters of the parameter (like, the first file
             is *.PID, etc.)
```

```
            ---- These fields do not require, but must be able
                 to accept, the extension.
            ---- The file titles on disk must end with the
                 specified extensions.
      -- Variables:  none
      -- I/O:
         --- Out:  TEXT file for output
             ---- Within ProjMod, "Out" is a GLOBAL VARIABLE and
                  is not passed through parameter lists to each
                  subroutine.
             ---- Where routines in ProjModU (this file) access
                  routines in ProjPrSt, "Out" is in the parameter
                  lists.
         --- files:  the routine will read the six files whose
             filenames are in the parameters
    - Action:  reads the six files, processes them to report the
      most probable identity of the target and the recommended
      action the fighter should take
    - Output:
      -- Parameters:
         --- ErrorExit:  TRUE iff the code found an error from
             which it cannot recover
         --- MostLikelyID:  the position in PIDFileTitle of the
             target identity most likely to represent the target
         --- RecommendAction:  the postition in PACFileTitle of
             the action with the highest expected utility given
             the information known about the target
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:  none
    - Last modified:  17 Jan 92 }

  PROCEDURE ProjMod
  ( PIDFileTitle,
    PACFileTitle,
    ESPFileTitle,
    ROWFileTitle,
    PRRFileTitle,
    USFFileTitle: TFullString;
    VAR ErrorExit: BOOLEAN;
    VAR MostLikelyID,
        RecommendAction: INTEGER );


    { ----- PROCEDURE AssessActions -----
      - Input:
        -- Parameters:
           --- HighIdent:  highest identity number being worked
           --- HighAct:  the high-numbered action
           --- LocalUtils:  a matrix, though internally organized
               as a singly-dimensioned array.  Stores a utility
               value for each combination of identities and
               actions.
           --- DistributionOfID:  a vector giving a probability
               distribution for the possible identities
        -- Variables:
```

```
      --- LowAct
      --- LowIdent
    -- I/O:  none
  - Action:  performs a vector inner product between
    DistributionOfID (on one hand) and each of a series of
    vectors in LocalUtils (on the other hand), each of which
    represents the utilities of a given action for each
    possible identity.  The result of each vector inner
    product is an expected utility; the result of the series
    of vector inner products is a vector of expected
    utilities, one for each action.  The routine reports the
    position of the maximum of those as RecommendAction.
  - Output:
    -- Parameters:
      --- RecommendAction:  the vector position of the
          action with the highest expected utility
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:  none
  - Last modified:  31 Jan 92 }


PROCEDURE AssessActions
( HighIdent: RangeID;
  HighAct: RangeAct;
  VAR LocalUtils: TLocalUtils;
    { - VAR to save memory; used only for input }
  VAR DistributionOfID: TDistributionOfID;
    { - VAR to save memory; used only for input }
  VAR RecommendAction: INTEGER );

VAR
  LocAct: RangeAct;
  LocID: RangeID;
  LUIndex: RangeUtils;
  ExpUtil: TDistributionOfID;

BEGIN { - PROCEDURE AssessActions }

  { - initialize }
  LUIndex := LowUtilNum;

  { - compute the expected utility vector }
  FOR LocAct := LowAct TO HighAct
  DO
    BEGIN

      ExpUtil [ LocAct ] := 0;
      FOR LocID := LowIdent TO HighIdent
      DO
        BEGIN

          ExpUtil [ LocAct ] := ExpUtil [ LocAct ] +
            DistributionOfID [ LocID ] *
            LocalUtils [ LUIndex ];
          INC ( LUIndex );
```

```
        END; { - FOR LocID := LowIdent TO HighIdent }

    END; { - FOR LocAct }

  { - find the highest expected utility }
  RecommendAction := LowIdent;
  FOR LocID := (   LowIdent + 1   ) TO HighIdent
  DO
    IF ( ExpUtil [ LocID ] > ExpUtil [ RecommendAction ] )
    THEN
      RecommendAction := LocID;

  { - RecommendAction is 0-based; convert to 1-based }
  INC ( RecommendAction );

END; { - PROCEDURE AssessActions }


{ ----- PROCEDURE IdentifyTarget -----
  - Input:
    -- Parameters:
       --- HighIdent:  the highest identity number being
           worked
       --- ROWVector:  a vector representation of the sensor
           reports coming in
       --- Arc, Node:  the data structures from PRNTHRM UNIT
           which store a probability ratio net.  The net
           stored here reflects relative probabilities of
           identies and reports about the world.
    -- Variables:
       --- LowIdent
       --- LowNodeNum
    -- I/O:  none
  - Action:  This routine uses tools in the PRNTHRM UNIT to
    reorganize the probability ratio net in Arc and Node.
    The desired output is a net which gives probabilities of
    identity given the report of the world.
  - Output:
    -- Parameters:
       --- Arc, Node:  a reorganized probability ratio net
       --- ErrorExit:  TRUE iff the program encounters an
           error it could not recover from
       --- MostLikelyID:  the vector position of the identity
           most likely to be associated with the target,
           given the report of the sensors
       --- DistributionOfID:  a probability distribution
           giving the probability of each identity for the
           target with the given report of the world
    -- Variables:  none
    -- I/O:  none
  - External subroutines used:
    -- DirectArcsToNode
    -- FlattenPRN
  - Last modified:  31 Jan 92 }
```

```
PROCEDURE IdentifyTarget
( HighIdent: RangeID;
  VAR Arc: TArc;
  VAR Node: TNode;
  VAR ErrorExit: BOOLEAN;
  VAR MostLikelyID: INTEGER;
  VAR DistributionOfID: TDistributionOfID );


  { ----- PROCEDURE FormTheDistribution -----
    - Input:
      -- Parameters:
         --- HighIdent:  the high identity number
         --- Arc, Node:  PRN
      -- Variables:
         --- LowArcNum
         --- LowIdent
      -- I/O:  none
    - Action:  using information in the PRN, forms the
      distribution on the identities
    - Output:
      -- Parameters:
         --- Arc, Node:  updated PRN
         --- DistributionOfID:  the distribution on the
             identities
      -- Variables:  none
      -- I/O:  none
    - External subroutines used:  none
    - Last modified:  2 Feb 92 }

  PROCEDURE FormTheDistribution
  ( HighIdent: RangeID;
    VAR Arc: TArc;
    VAR Node: TNode;
    VAR DistributionOfID: TDistributionOfID );

  VAR
    LocArcNum: TArcNum;
    LocTail: TNodeNum;
    LocID: RangeID;
    Sum: REAL;

  BEGIN { - PROCEDURE FormTheDistribution }

    { - collect the relevant probability ratios }
    FOR LocArcNum := LowArcNum TO    Arc . NArcs
    DO
      BEGIN
        LocTail :=   Arc . Tail [ LocArcNum ];
        IF (   Node . NodeLabel [ LocTail ]   <> '' )
        THEN
          BEGIN { - the tail is relevant }
            LocID :=
              ORD (   Node . NodeLabel [ LocTail ] [ 1 ]   );
            DistributionOfID [ LocID ] :=
              Arc . Weight [ LocArcNum ];
```

```
          END; { - IF ( tail is relevant ) }
        END; { - FOR LocArcNum }

      { - convert the ratios to a distribution }
      Sum := 1;
      FOR LocID := (   LowIdent + 1   ) TO HighIdent
      DO
        Sum := Sum + DistributionOfID [ LocID ];
      DistributionOfID [ LowIdent ] :=   1 / Sum;
      FOR LocID := (   LowIdent + 1   ) TO HighIdent
      DO
        DistributionOfID [ LocID ] :=
          DistributionOfID [ LowIdent ] *
          DistributionOfID [ LocID ];

    END; { - PROCEDURE FormTheDistribution }


    { ----- PROCEDURE MsgErrorExit -----
      - Input:
        -- Parameters:
           --- ProcName:  the name of the procedure which set
               ErrorExit to TRUE
        -- Variables:  none
        -- I/O:  none
      - Action:  puts error messages on the screen
      - Output:
        -- Parameters:  none
        -- Variables:  none
        -- I/O:  screen messages
      - External subroutines used:
        -- PressAnyKey
      - Last modified:  31 Jan 92 }

    PROCEDURE MsgErrorExit
    ( ProcName: STRING );

    BEGIN { - PROCEDURE MsgErrorExit }
      WRITELN ( Out );
      WRITELN ( Out );
      WRITELN ( Out );
      WRITELN ( Out , 'Error termination' );
      WRITELN ( Out );
      WRITELN ( Out ,
        'ErrorExit became TRUE after a call to ' , ProcName );
      PressAnyKey;
    END; { - PROCEDURE MsgErrorExit }



VAR
  LocID: RangeID;
  LocNodeNum: TNodeNum;
  NArcToPoint: INTEGER;
  ArcToEmbed,
  ReplaceArc: TArcNum;
```

```
  BEGIN { - PROCEDURE IdentifyTarget }

    FlattenPRN ( Arc , Node , ErrorExit );
    IF ( ErrorExit )
    THEN
      BEGIN
        MsgErrorExit ( 'FlattenPRN' );
        EXIT; { - from IdentifyTarget }
      END; { - IF ( ErrorExit ) }

    { - find the node for the first identity }
    LocNodeNum := LowNodeNum;
    WHILE
      (   LENGTH (   Node . NodeLabel [ LocNodeNum ]   )   =
        0 ) OR
      (   Node . NodeLabel [ LocNodeNum ] [ 1 ]   <>
        CHR ( 0 )   )
    DO
      INC ( LocNodeNum );

    { - point all nodes at the one for the first identity }
    DirectArcsToNode ( LocNodeNum , NArcToPoint , ErrorExit ,
      Arc , Node );
    IF ( ErrorExit )
    THEN
      BEGIN
        MsgErrorExit ( 'DirectArcsToNode' );
        EXIT; { - from IdentifyTarget }
      END; { - IF ( ErrorExit ) }

    FormTheDistribution ( HighIdent , Arc , Node ,
      DistributionOfID );

    { - find the max probability }
    MostLikelyID := LowIdent;
    FOR LocID := (   LowIdent + 1   ) TO HighIdent
    DO
      IF ( DistributionOfID [ LocID ] >
           DistributionOfID [ MostLikelyID ] )
      THEN
        MostLikelyID := LocID;

    { - MostLikelyID is 0-based; convert it to 1-based }
    INC ( MostLikelyID );

  END; { - PROCEDURE IdentifyTarget }


VAR
  HighIdent: RangeID;
  HighAct: RangeAct;
  ROWVector: TROWVector;
  LocalUtils: TLocalUtils;
  Arc: TArc;   { - TArc declared in PRNTHRM UNIT }
  Node: TNode; { - TNode declared in PRNTHRM UNIT }
```

```
    DistributionOfID: TDistributionOfID;

  BEGIN { - PROCEDURE ProjMod }

    { - initialize }
    ErrorExit := FALSE;
    MostLikelyID := 0;     { - in case of error exit }
    RecommendAction := 0; { - in case of error exit }

    ProcessStarter (
      { - in and out } PIDFileTitle , PACFileTitle ,
                       ESPFileTitle , ROWFileTitle ,
                       PRRFileTitle , USFFileTitle ,
                       { - though the above are altered, the
                           altered value is not used; this
                           arrangement saves a little memory
                           during execution }
      { - outputs    } Out , ErrorExit , HighIdent , HighAct ,
                       ROWVector , LocalUtils , Arc , Node );
    IF ( ErrorExit ) THEN
      EXIT; { - from ProjMode }

    IdentifyTarget (
      { - inputs     } HighIdent ,
      { - in and out } Arc , Node ,
      { - outputs    } ErrorExit , MostLikelyID ,
                       DistributionOfID );
    IF ( ErrorExit ) THEN
      EXIT; { - from ProjMode }

    AssessActions (
      { - inputs     } HighIdent , HighAct , LocalUtils ,
                       DistributionOfID ,
      { - outputs    } RecommendAction );

  END; { - PROCEDURE ProjMod }


BEGIN { - of initialization code for ProjModU }
  ASSIGN ( Out , '' ); { - standard output;
                         supports redirection }
  REWRITE ( Out );
END.
```

Bibliography

ATRWG Tech Comm. — Advanced Target Recognizer Working Group Technology Committee. *Comparison of Statistical Pattern Recognition, Model-Based and Neural Network Approaches for Automatic Target Recognizer.* Internal white paper, sponsored by the Defense Advanced Research Projects Agency, The Department of Defense Working Group on Advanced Target Recognition, Joint Guidance and Control Committee, 10 October 1991.

Bajcsy 88. — Bajcsy, Ruzena. "Perception With Feedback," *Proceedings of the Image Understanding Workshop*, Volume I. 279–287. Defense Advanced Research Projects Agency, Information Science and Technology Office, April 6–8, 1988.

Turbo Pascal. — Borland International, Inc. *Turbo Pascal Version 6.0 Programmer's Guide.* Scotts Valley, California: Borland International, Inc., 1990.

NY Times, 2 May 89. — Browne, Malcolm W. "Is That a Grimace? Ask a Pigeon," *New York Times*, 2 May 1989, p. C7.

IEEE Spectrum, Aug 87. — Christiansen, Donald. "Pigeon Guided Missiles That Bombed Out", *IEEE Spectrum*, August 1987, p. 46.

NY Times, 20 Aug 88. — Cushman, John H., Jr. "11 Minutes to Downing of an Airliner," *New York Times*, 20 August 1988, p. 5.

Aerospace Amer, Feb 90. — DeMeis, Richard. "JSTARS in the European Skies", *Aerospace America, 28*: 26–28, 30 (February 1990).

Dodging Friendly Fire. — "Dodging Friendly Fire," *Time*: 24 (18 Feb 91).

Gaebler. — Gaebler, Matthew E., pilot for American Airlines and a USAF A–10 pilot until 1990. Huber Heights, Ohio, 15 February 1992.

Howard 83. — Howard, Ronald A. "The Evolution of Decision Analysis", *Readings on the Principles and Applications of Decision Analysis*, Volume I, edited by Ronald A. Howard and James E. Matheson. Menlo Park, California: Strategic Decisions Group, 1983, pp. 5–16.

Howard 89. — Howard, Ronald A. "Knowledge Maps," *Management Science, 35*: 903-922 (August 1989).

AF Magazine, Jun 91. Grier, Peter. "Joint STARS Does Its Stuff", *Air Force Magazine*: 38–42 (June 1991).

Kabrisky. Kabrisky, Dr. Matthew, pattern recognition researcher and member of AFIT faculty. Wright-Patterson AFB, Ohio, 27 February 1992.

Final Report 91. McDonnell Aircraft Company. *Phase I Final Report of the Pilot's Associate Program.* Wright Laboratory Technical Report WL–TR–91–7006. Wright-Patterson AFB Ohio: Wright Laboratory, September 1991.

Morlan 91. Morlan, Bruce W. Unpublished dissertation, 1991.

Morlan. Morlan, Maj Bruce W., developer of Probability Ratio Nets. Wright-Patterson AFB Ohio, multiple conferences throughout thesis project.

Powers and others 84. Powers, Michael J. and others. *Computer Information Systems Development: Analysis and Design* . Cincinnati: South-Western Publishing Company, 1984.

Pratt and others 64. Pratt, John W. and others. "The Foundations of Decision Under Uncertainty: An Elementary Exposition," *Journal of the American Statistical Association, 59*: 353-375 (June 1964).

Quinlan 90. Quinlan, J. R. "Decision Trees and Decisionmaking," *IEEE Transactions on Systems, Man, and Cybernetics, 20*: 339-346 (March/April 1990).

Raiffa 68. Raiffa, Howard. *Decision Analysis.* Reading, Massachusetts: Addison-Wesley, 1968.

Schoemaker 82. Schoemaker, Paul J. H. "The Expected Utility Model: Its Variants, Purposes, Evidence, and Limitations," *Journal of Economic Literature, XX*: 529-563 (June 1982).

Singstock 91. Singstock, 2d Lt Brian D. *Infrared Target Recognition* . MS thesis, AFIT/GE/ENG/91D-49. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB Ohio, December 1991.

Stieven. Stieven, Major John S., U.S. Air Force F-4 pilot. Wright-Patterson AFB Ohio, 19 February 1992.

Tarr 91. Tarr, Capt Greg. *Multi-Layerd Feedforward Neural Networks for Image Segmentation.* PhD dissertation.

School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB Ohio, December 1991.

Tatman and Shachter 90. Tatman, Joseph A. and Ross D. Shachter. "Dynamic Programming and Influence Diagrams," *IEEE Transactions on Systems, Man, and Cybernetics, 20*: 365-379 (March/April 1990).

White 90. White, Chelsea C. "A Survey on the Integration of Decision Analysis and Expert Systems for Decision Support," *IEEE Transactions on Systems, Man, and Cybernetics, 20*: 358-364 (March/April 1990).

Zelnio 91. Zelnio, Edmund G. "ATR [automatic target recognizer] Paradigm Comparison with Emphasis on Model-Based Vision." Keynote paper at Model-Based Vision Development and Tools Conference. Boston, Massachusetts, 14–15 November 1991. Proceedings to be published.

Vita

Major Garrison H. Flemings was born 22 December 1954. He graduated from McGuffey High School near Claysville, Pennsylvania in May 1972. He earned a commission in the United States Air Force and graduated from the United States Air Force Academy in Colorado Springs, Colorado on 2 June 1976, having earned a Bachelor of Science in Mathematics. Upon completion of Undergraduate Pilot Training, he flew the B-52G at Loring AFB, Maine. Later, he flew the B-52G at Mather AFB, California. He served three years on the faculty of the Squadron Officer School at Maxwell AFB, Alabama. He completed a Master of Science in Computer and Information Science with Troy State University, Montgomery in Montgomery, Alabama in 1986. He graduated from Air Command and Staff College at Maxwell AFB, Alabama in 1988. After two years flying the B-52G at Eaker AFB, Arkansas, he entered the Strategic and Tactical Sciences program in the School of Engineering, Air Force Institute of Technology (AFIT). The Air Force selected him for promotion to Lieutenant Colonel during his student time at AFIT. He has been married to Kathryn E. (Haynes) Flemings, formerly of Independence, Missouri, since 1980. They have two children, Jennifer (age 9) and Richard (age 6).

Permanent address: 1416 Downers Place
Independence, Missouri
64056

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1992	Master's Thesis

4. TITLE AND SUBTITLE
A DECISION-THEORETIC APPROACH TO RECOMMENDING ACTION IN THE AIR-TO-GROUND AIRCRAFT OF THE FUTURE

5. FUNDING NUMBERS

6. AUTHOR(S)
Garrison H. Flemings, Major, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology,
WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GST/ENS/92M-03

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint Cockpit Office
Systems Development Branch
Wright Laboratory
Wright-Patterson AFB OH 45433

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Designers of air-to-ground fighters of the future should consider including decision support systems to relieve pilots of some of the present workload. Those decision support systems should use utility theory. This work reviews two techniques of utility theory (decision trees and influence diagrams) and a new technique due to Morlan (probability ratio nets). The work includes a proposed structure for action recommendation systems in air-to-ground fighters of the future. The work describes an implementation of probability ratio nets. The work demonstrates the value of decision theory to air-to-ground fighters of the future by way of solving an example problem and several variation of it.

14. SUBJECT TERMS
Utility Theory; Pattern Recognition; Game Theory; Statistical Decision Theory

15. NUMBER OF PAGES
182

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency. Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in .... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availablity Statement. Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- DOD - Leave blank
DOE	- DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA	- NASA - Leave blank
NTIS	- NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.