UCLA-ENG-REP-7666
JUNE 1976

UNIVERSITY OF CALIFORNIA

Los Angeles

An Interactive Program for Conversational Elicitation of Decision Structures

A dissertation submitted in partial satisfaction of
the requirements for the degree of Doctor of Philosophy
in Engineering

by

Antonio Leal

1976

The dissertation of Antonio Leal is approved.

Norman C. Dalkey

Edward C. Carterette

Joseph A. Goguen Jr.

Judea Pearl

Committee Chairman

University of California, Los Angeles

1976

To my wife,

Mary F. Leal

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor and committee chairman, Dr. Judea Pearl, of the Engineering Systems Department for his supervision, guidance, and instruction. His keen insight and sharp criticisms were invaluable. I am most grateful for his patience and understanding. I would also like to thank the members of the graduate committee for their helpful comments, suggestions, and approval: Dr. Norman C. Dalkey, Engineering Systems Department; Dr. Edward C. Carterette, Psychology Department; Dr. Joseph A. Goguen, Computer Science Department; and Dr. James S. Dyer, Graduate School of Management.

I extend my gratitude to Ward Edwards, Social Sciences Research Institute, University of Southern California, Los Angeles, for spending his time in discussions with me. I would like to thank Dr. Gerald Shure, Dr. Ralph Meeker, and Dr. Miles Rogers of the Center for Computer-Based Behavioral Studies (CCBS) for their support on the PDP-10 computer system and to Craig M. Rogers for his help with my programming problems. Finally, a special thanks to Dr. Robert Young of the Advanced Research Projects Agency (ARPA) for his interest and computational support. This research was funded in part by the National Science Foundation Grant GJ-42732.

VITA

December 9, 1942--Born, Chicago, Illinois

1965--B.A., University of Illinois

1966--M.S., University of Illinois

1966-1969--Project Manager, Computer Sciences Corporation, Washington, D.C.

1969-1973--Associate Research Scientist, System Development Corporation, Santa Monica, California

1975-Present--Senior Scientist, Perceptronics, Inc., Woodland Hills, California

ABSTRACT OF THE DISSERTATION

An Interactive Program for
Conversational Elicitation
of Decision Structures

by

Antonio Leal
Doctor of Philosophy in Engineering
University of California, Los Angeles, 1976
Professor Judea Pearl, Chairman

An interactive computer program has been designed and implemented that elicits a decision tree from a decision maker in an English-like conversational mode. It emulates a decision analyst who guides the decision maker in structuring and organizing his knowledge about a particular problem domain. The objectives of the research are: (1) to provide the decision analysis industry with a practical automated tool for eliciting decision structures where manual elicitation techniques are either infeasible or uneconomical, (2) to cast the decision analyst's behavior into a formal framework in order to examine the principles governing the elicitation procedure and gain a deeper understanding of the analysis process itself, and (3) to provide experimental psychologists with an

automated research tool for coding subjects' perception of problem situations into a standard and formal representation.

The approach centers on the realization that the dynamic process of decision tree elicitation is almost identical to conducting a heuristic search on game trees. Heuristic search techniques, when applied to tree elicitation, permit real-time rollback and sensitivity analysis as the tree is being formulated. Thus, it is possible to concentrate effort on expanding those parts of the tree which are crucial for the resolution of the solution plan. The program requires the decision maker to provide provisional values at each intermediate stage in the tree construction, that estimate the promise of future opportunities open to him from that stage. These provisional values then serve a role identical to a heuristic evaluation function in selecting the next node (scenario) to be explored in more detail.

The program is completely domain independent. That is, it assumes no prior knowledge specific to any problem environment. Through interaction with a prescribed sequence of queries, the decision maker is able to supply information on the structuring of his particular decision predicament. For example, the decision maker may supply information about the various decision alternatives representing possible future opportunities, outcomes of uncertain events along with likelihood relations among them, and immediate costs, if any. In addition, he is permitted to make a mental "experiment" that, when actually performed at a later time, will sharpen the likelihood estimates of future events. The final result of the computer interview is a complete "solution plan" that recommends an action for all anticipated contingencies.

I. INTRODUCTION

Decision analysts are often called upon to assist in the solution of planning problems ranging over a large variety of domains. In such circumstances, the decision analysts usually possess less specific knowledge about the problem domain than their customers and their contributions are confined primarily to formalization of the problem and optimization of the solution. While optimization is usually performed on electronic computers, formalization has invariably been accomplished manually, using lengthy interviews with persons intimately familiar with the problem domain.

This dissertation describes an attempt to automate the formalization phase using an interactive computer system that guides the decision maker through a structured English-like dialog and constructs a decision tree from his responses. The objectives of this work are three-fold: (1) to provide the decision analysis industry [1,2] with a practical automated tool for eliciting decision trees in cases where manual elicitation techniques are either infeasible or uneconomical, (2) to cast the decision analyst's behavior into a formal framework in order to examine the principles governing the elicitation procedure and gain a deeper insight into the analysis process itself, and (3) to provide experimental psychologists with an automated research tool for encoding subjects' perception of problem situations into a standard and formal representation.

The absence of an established formal procedure of eliciting decision trees is not difficult to understand. Tree construction is the first step in the decision-making endeavor: the formal representation

of informal concepts [3]. Since the informal concepts reside in the decision maker's perception of the problem environment, the translation process consists of discussions and interviews as well as attempts to educate him as to the type of information he is to supply. It often requires a special insight and ingenuity on the part of the analyst to direct the conversation and phrase the queries in a way that will yield both informative and reliable information.

From a practical viewpoint, the major drawback of manual interviews is their length and cost. Since real-time analysis of decision trees is beyond the limitation of human computational capability, it invariably happens that many hours of interviews are spent on eliciting portions of the decision tree that do not have decisive bearing on the problem at hand. This fact can only be discovered at a later stage, when the problem structure is formalized and when a sensitivity analysis has been conducted on a computer. During the interview itself, however, it is impossible for the analyst to process the entire information obtained by him up to that point and to select the optimum course for conducting his future inquires. For example, in eliciting utilities it is a common practice to extract indifference curves among several value attributes. Often the decision analyst is forced to elicit these curves over a wide range of attributes only to find out later that eliciting preferences among a few selected points would have been sufficient to determine the entire problem [4]. Similar situations occur in the process of eliciting conditional probabilities and in the expansion of very complex trees. Thus, our inability to perform real-time analysis of the information at hand forces us to waste precious time on inconsequential, detailed queries.

A direct man-machine interface could provide three distinct advantages. First would be the capability of real-time sensitivity analysis, which in turn could be used to guide the growth of the decision tree in only the most promising directions. Detailed queries could be reserved, then, for those branches of the tree which, on the basis of the information obtained thus far, seem most crucial to the main decision-related goals.

The second advantage of direct man-machine elicitation would be the ease of updating the system with new knowledge. It is expensive (if not impractical) in many situations to solicit the assistance of a decision analyst each time the decision maker gains a new piece of knowledge. A conversational system, on the other hand, could provide an intimate medium that could be updated quickly even by the non-technical planner.

The third advantage would lie in the feasibility of incorporating real-time Delphi methods for aggregating the opinions of several experts [5]. Decision structures elicited from other members of the team could be interrogated at will and displayed during the elicitation process to help an expert enrich or revise his previous structure. Disagreement could be detected, isolated, and brought up for further team discussion.

The art of directing a tree-eliciting dialog is governed by two conflicting goals. On one hand, the analyst attempts to bring to bear the most complete knowledge that the decision maker may possess relevant to his current problem. On the other hand, he desires to do so with the least number of queries. Cutting down on the number of queries

could only be accomplished at the expense of reducing the reliability of the information obtained. The analyst could, of course, limit the queries to holistic, global judgments, avoiding the painstaking detailed queries or he may terminate the tree prematurely, ask for value judgments on the terminal nodes, and begin the optimization phase. Such schemes, however, would defy the very purpose of decision analysis.

Decision analysis is founded on the paradigm that people possess reliable procedures for detecting, storing, and retrieving fragments of knowledge, but possess much less reliable procedures for aggregating these fragments into a global inference. If it were not for this deficiency, there would be no purpose served by analysis. It would then be sufficient to ask the decision maker, "Which alternative action seems most valuable to you?" and when he responds, advise him, "Do it!" The fact that the analyst refrains from asking direct value judgments on actions but prefers to construct them mechanically from judgments on events, circumstances, and surmises, reflects the analyst's hope that mechanically constructed judgments, using Bayes' rollback procedures [6] are more faithful to the decision maker's experience than direct holistic judgments internally processed by him [7]. Thus, by asking more detailed questions and expanding the depth of the decision tree, the analyst expects to obtain a *refinement* of less accurate judgments that could not otherwise be obtained at an earlier stage. It is this tradeoff between the cost of each query and its contribution to the overall judgment accuracy that underlies the style that analysts select in conducting their dialogs.

This tradeoff bears a striking similarity to that which governs tree expansions in Artificial Intelligence procedures. Game playing, robot planning, and theorem proving programs [8] all seek to obtain close-to-optimal solutions without exhaustively searching through the underlying problem trees. In these applications, tree pruning is achieved via the use of a heuristic function: a simple rule, externally provided by the analyst or programmer, that computes a crude estimate of the value (strength or promise) of any tree node in reaching the goal. In the game of chess, for example, such a rule may prescribe the way in which the various aspects of any board configuration (i.e. material advantage, mobility, number of threatened pieces, etc.) should be combined to give a rough estimate of the overall strength of that position. The true value of each configuration (i.e. "win", "draw", or "lose") cannot, of course, be determined exactly unless the game tree is expanded exhaustively out to its terminal positions and a mini-max rollback procedure is applied [9]. Since such a search is utterly impractical, the strength of each position is determined by its rollback value after the heuristic evaluation function is assigned to the terminal nodes of a truncated tree. The strength could, of course, be determined by direct application of the heuristic function to the position in question. However, such evaluation would be far less reliable than one obtained by rollback over several levels of look-ahead. Thus, the purpose of tree expansion is, as in the case of decision trees, to obtain a more reliable estimate of a node's value. The value produced by the heuristic function may thus be considered to consist of the "true" value plus an error factor or noise that is reduced by tree expansion.

The availability of a heuristic evaluation function is one of the advantages that heuristic search has compared to decision tree expansion. The function is available *at each node* during expansion to estimate its relative potential for achieving the search objective. Thus, not only is this measure used to evaluate the tip nodes before rollback, but it also determines the order of node expansion. Heuristic search procedures select for expansion that node which, on the basis of the provisional evaluation function, seems most likely to separate the top contending plans. Using this technique, substantial savings in node expansion have been achieved, even with very crude heuristic functions.

In contrast, in manual decision tree elicitation, the analyst usually structures the entire tree (to some comfortable depth) before values are placed on the tip nodes. Drawing an analogy with heuristic search techniques, it seems that decision tree analysis could be enhanced considerably if values were available on nodes as the tree is being expanded. What would such values represent? They would be no different than values placed at the tip nodes of the completed tree: estimates of the relative utility of the outcomes. This utility represents a mental estimate of the rollback value of a complete theoretical tree emanating from that particular node on. It can, as before, be regarded as the final expected value with errors due to deficiencies in mental aggregation procedures. Nevertheless, if the decision maker is requested to provide values for nodes as they are being expanded, the information can be used to determine a node

expansion order. As the tree expands, these provisional values become more "refined", that is, closer to the true (mechanically processed) value.

The major objective of the elicitation program discussed here is the construction of the decision tree. While many other computer systems focus on the elicitation of utilities and probabilities [10], these aspects are not emphasized here. The decision maker is assumed to be able to provide value and likelihood estimates in numerical form, eliminating the need for construction from a sequence of binary choice queries. More sophisticated techniques for utility and probability elicitation could be incorporated into the program once the elicitation skeleton is developed.

The first portion of this dissertation deals with the fundamental concepts incorporated into the elicitation program. The latter portion describes the operational program itself. Section II describes heuristic search in more detail than discussed thus far and relates it to the node expansion algorithm imposed by real-time sensitivity analysis. In section III, the opportunity to gain information through experimentation is discussed and the method by which this opportunity is made available to the decision maker. The main elicitation procedure, as experienced by the decision maker using the program, is described in section IV along with typical queries. Section V demonstrates the operation of the program using a typical dialog with a decision maker facing a hypothetical problem situation. The internal structure and organization of the actual program is illustrated in section VI with a description of major

routines and their relation to the entire system. Section VII summarizes major conclusions of the research including a discussion of the advantages and deficiencies discovered during the development. A complete program code listing is supplied in the Appendix.

II. NODE EXPANSION ORDER

Traditional decision tree elicitation techniques require the decision maker to formulate decision alternatives and expected events as a first step toward tree construction. Only after the tree structure has been tentatively solidified, is the decision maker expected to place probabilities on event outcomes and values on projected future situations. Probabilities are placed at each internal event node and values are placed at each terminal ("tip") node of the tree. Each tip node represents a projected future situation (scenario) that is uniquely determined by the path from the initial decision (root) node to the tip of the tree. The decision maker must evaluate each individual path and place a numeric value on it in terms of a previously selected utility scale. After all values and probabilities have been determined, a computational (rollback) analysis determines which of the initial decision alternatives is the most favorable for the decision maker to take.

The current research emulates an "automated decision analyst" who, aside from recommending optimal action plans, also constructs the decision tree itself. It calls for a rational procedure to govern the growth of the decision tree. The adopted approach to such a procedure rests on the hypothesis that expanding the decision tree during an elicitation interview is equivalent to conducting a heuristic search in a complex problem tree. As an example of problem solution by search, consider the task of finding

the shortest route between two cities, say Los Angeles and Houston. In this domain, the search "tree" is a representation of possible routes from Los Angeles to Houston and a "node" represents a highway junction. A search procedure would investigate each route, one at a time, in an attempt to locate the optimal path. It is, essentially, problem solution by exhaustive enumeration. The final result is an optimal solution path that recommends a course of action at each node. In this case, the result would be a list of "directions" which specify a highway to follow at each junction.

The game of chess provides another example of a problem which can be attacked by search techniques. In this case, however, it is not a solution path that is required, but a solution *plan*. Each node in the search tree represents a different chess board configuration. The nodes are connected by the legal moves of the game. That is, the transition from one node (board position) to another in the search tree is governed by the possible chess moves that the player can make at that time. Since the moves of the opponent are unknown, a solution plan must recommend an optimal move regardless of his actions. Such a plan is an optimal "subtree" contained in the original tree of all possible moves.

The search procedure itself begins at the initial situation, called the "root" node, and generates possible successor nodes according to the characteristics of the specific problem domain. In the cities example, this corresponds to locating each highway junction leading out from Los Angeles. In chess, each possible move

from the current position is considered. Once this generation is accomplished, the node is said to have been "expanded". By repeated expansion, longer and longer paths are generated and the search tree is "grown".

Although the expansion rules follow straight-forwardly from the problem environment, the *order* of expansion remains to be defined. That is, there are no intrinsic rules that select the next node to be expanded. Two well-known tree search procedures are available that search the nodes in a uniform or "blind" manner. They are "breadth-first" and "depth-first" expansion [8]. Uniform node expansion algorithms generate an expansion order that is independent of any information about the relationship between the candidate nodes and the solution. Breadth-first expansion grows the tree on equal-depth contours (see Figure 2.1). All nodes in the outer-most layer are expanded before any node in deeper layers. This insures that all paths from the root have equal length. In the context of decision tree elicitation, for instance, a breadth-first expansion order means that each scenario has an equal level of refinement. Thus, tree growth will, in general, follow an orderly progression in time and the decision maker will not be forced to make predictions in a widely varying time domain. The disadvantage is, of course, that he is forced to elaborate non-crucial scenarios.

The second uniform expansion algorithm is "depth-first" (see Figure 2.2). Each path is explored to its logical conclusion (or to a prespecified depth bound) and then related paths are generated in an overall left-to-right pattern. In the context of

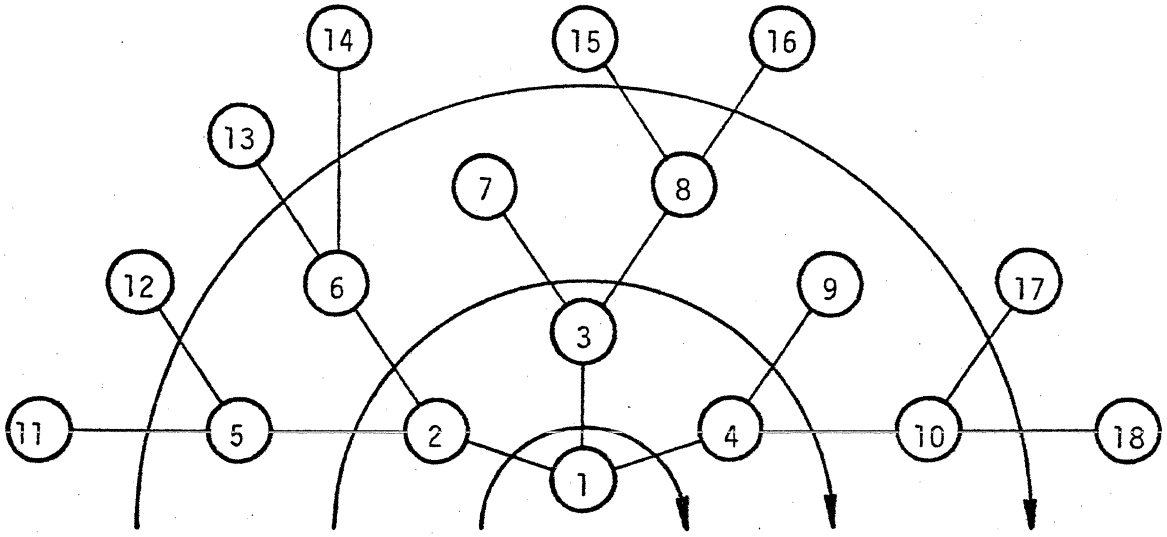


Figure 2.1. Breadth-First Expansion Order

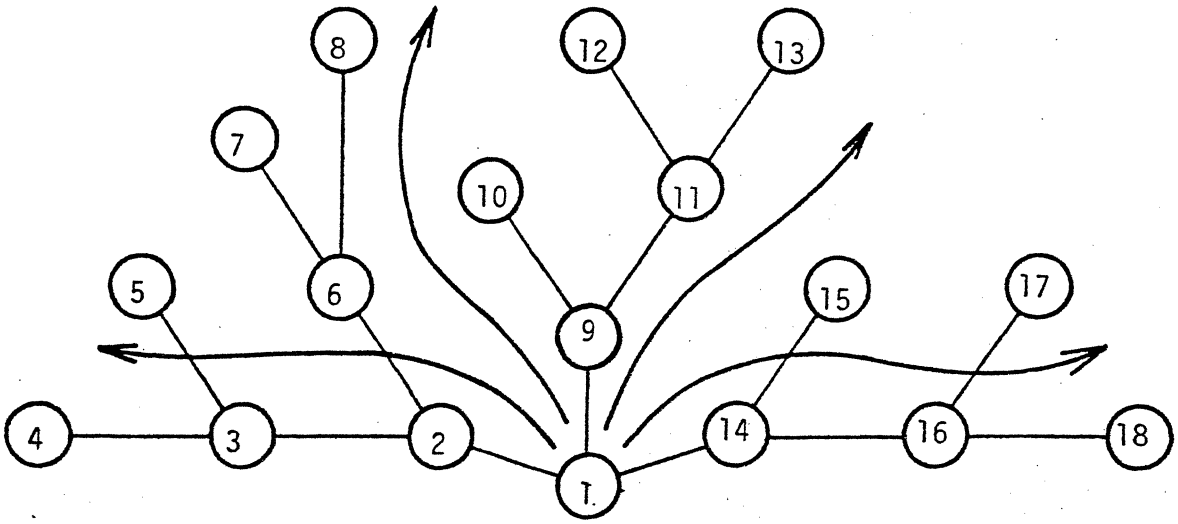


Figure 2.2. Depth-First Expansion Order

decision analysis, the decision maker would be asked to follow a scenario to its conclusion, however, precious time could, again, be wasted in exploring scenarios that have low impact on discovering the solution.

In contrast to the uniform expansion algorithms described above, heuristic search techniques attempt to use information about the relationship between candidate nodes and the solution. Such information is in the form of a "heuristic function" which is a rule for estimating the difficulty of reaching the solution from any given node. In the cities example, a heuristic may be the direct air distance from any highway junction to Houston (if known). In chess, a heuristic would estimate the difficulty of achieving a checkmate from any given game configuration. The numeric value of the heuristic has the effect of ordering the candidate nodes as to their promise in finding a solution. That node with the most promise can be selected as the next to expand.

A "perfect heuristic"* should produce a value of 1 for all nodes in the optimal solution plan and a value of 0 for all other nodes. In this case, there would be no "search" since a perfect strategy would exist for accomplishing the objective. In real-world situations, however, we must usually be satisfied with a less-than-perfect heuristic function and a finite-horizon search tree. This

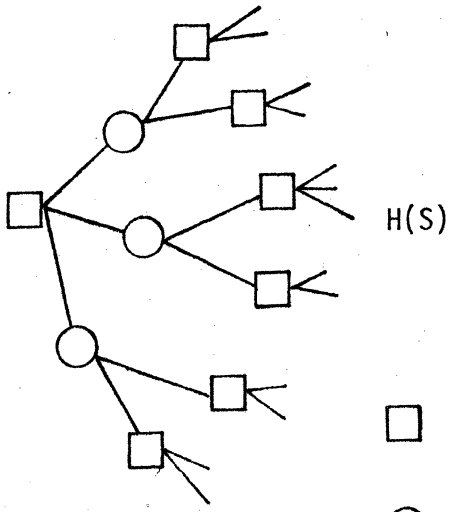
* A "perfect heuristic" is somewhat of a contradiction in terms, since the connotation of the word "heuristic" is a "rule of thumb" that comes from human intuition.

will cause some nodes to be expanded that do not lie in the optimal solution plan because the heuristic function will provide a non-optimal ordering of the candidate tip nodes.

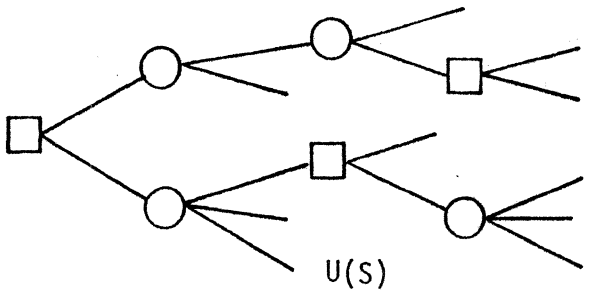
Figure 2.3 compares heuristic search with decision tree expansion. The method used to direct the interview requires the decision maker to estimate *provisional* values that represent opportunities open to him by anticipated action/event combinations.

The provisional values are used in the same way as the heuristic function in tree searching. The difference is that the heuristic function is initially supplied by the analyst or programmer and must be defined over the entire problem domain, while the provisional values are supplied by the decision maker during the elicitation interview. As a consequence, node expansions follow the decision maker's perception of event relationships in the environment rather than following predefined transition rules. These values can, as in the case of heuristic search, be considered to be the "true" values with error due to "noise". The amount of error is dependent on the size of the tree and the decision maker's ability to provide correct information. Since the decision maker's predictive ability remains relatively constant, the amount of error is more directly related to the size of the tree.

The order of expansion algorithm should be consistent with the overall objective of the decision analysis. Thus, the primary interest is not, as might be expected, to expand the most valuable (highest utility) path discovered, but rather, to expand that node



H(S)



U(S)

First player's turn to move

Second player's turn to move

Decision node

Event node

HEURISTIC SEARCH ON GAME TREES	DECISION TREE ELICITATION
• Object is to find the path (plan) with the highest heuristic value H(S) with the minimum number of node expansions.	• Object is to find the path (plan) with the highest utility U(S) with the minimum number of questions.
• Complete tree unavailable explicitly. (Implicitly contained in game rules.)	• Complete tree unknown to the analyst. (Resides in the decision maker's knowledge.)
• Expansion follows state transition rules. (Legal moves.)	• Expansion follows the decision maker's perception of event/action relationships.
• Heuristic function provided by analyst.	• Provisional values provided by decision maker.
• Heuristic function guides search.	• Provisional values determine next question.
• Mini-max rollback.	• Expecti-max rollback.
• Terminal nodes determined by rules(win/loss)	• Terminal nodes determined by decision maker.

Figure 2.3. Analog between Heuristic Search on Game Trees and Decision Tree Elicitation

which is *most likely* to change the currently best initial decision. If this particular node is not part of the subtree leading outward from the currently highest initial branch, then its value must be *incremented* enough to bring the value of the initial branch leading to that node up until it equals the value of the highest branch. Thus, the sensitivity measure consists of estimating the amount of change (differential) in a given provisional node value necessary to cause a change in the currently best initial decision.

For example, in Figure 2.4, a partial tree is shown with initial decision branches b_1, b_2, and b_3. Branch b_2 is shown with an expanded event node that has two outcomes A and B. Assume that from a previous rollback calculation, the values of the three decision branches are 5, 3, and 2 respectively. Thus, b_1 represents the currently most promising decision. To calculate the sensitivity differential of node A, the following question is posed, "How much should the value of node A (currently 5) be raised so that the value of the initial branch leading to node A (i.e. b_2) will equal the currently highest branch (i.e. b_1)?" Branch b_2 must obviously be incremented by at least 2, but node A is only contributing 20% of its value to it. Node A must be raised to a total higher than 15 in order to cause b_2 to exceed b_1. Thus, the sensitivity differential of node A is 10. Similarly, B must be raised to 5 (assuming no other changes) in order to cause b_2 to increase by 2. Since A must be raised more than B, the value of A is said to be more "robust" than that of B. The formula for the sensitivity

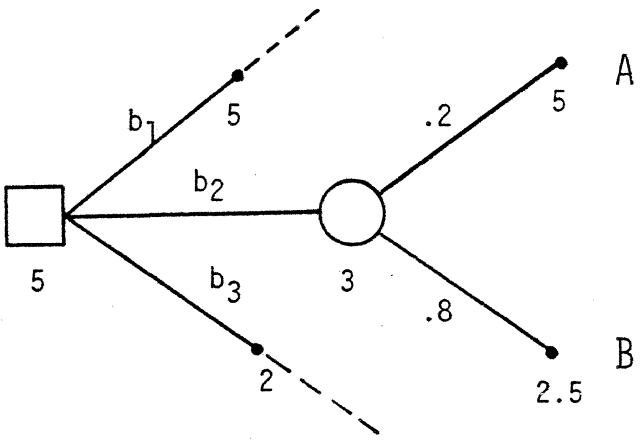


Figure 2.4. Basic Sensitivity Differential

differential is straight-forward. It can be shown for any node n, that if the path leading to n contains only event nodes, the sensitivity differential S(n) is:

$$(1) \qquad S(n) = \frac{\Delta b_n}{\pi P_n}$$

where Δb_n is the difference between the currently highest initial decision branch b_{max} and that decision branch b_n which leads to node n, and πP_n is the product of all the probabilities along the path from b_n to n. Nodes in the subtree leading out from b_{max} itself are simply compared to the second highest initial branch value to see how much they would affect b_{max} if they were *lowered*. A somewhat more elaborate computational procedure (described below) is required to calculate S(n) across a path containing both decision and event nodes.

The proof of equation (1) is as follows. Consider, temporarily, a decision tree consisting solely of event nodes except the root which is a decision node. Figure 2.5 is an example of such a tree. Let d be the root decision node and let n be an arbitrary tip node with a provisional value of V(n). Let e_n be that event node which is a successor to d and leads to the subtree containing n. Let e_{max} be the event node successor to d with the maximum rollback value. Every non-initial branch in the tree must have an associated probability ofoccurrence. Let $P(e_n,n) = P(e_n),..,P(n)$ be the sequence of probabilities from node e_n to node n along the entire path.

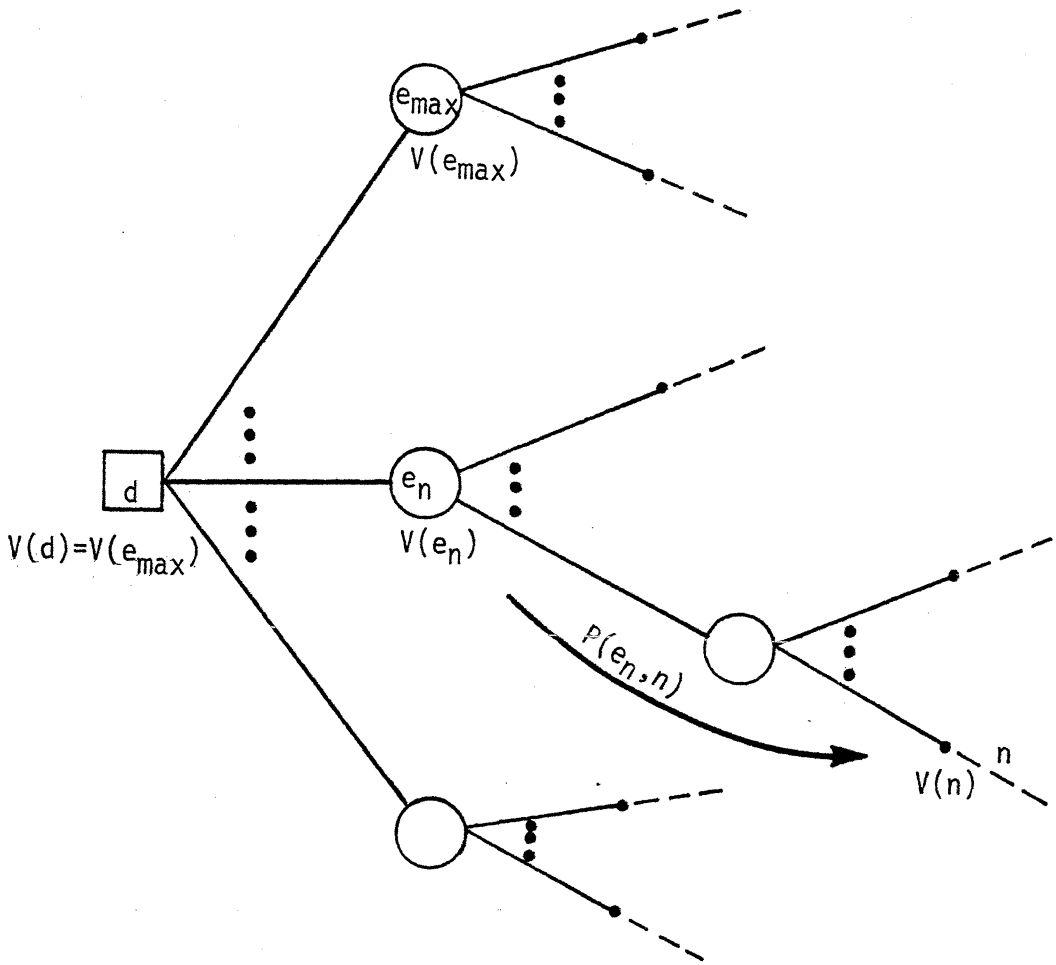


Figure 2.5. An Event Tree

Now, if the subtree following e_n is collapsed to a single level, the number of branches will equal the number of tip nodes in the original subtree. That is, every path in the original subtree will be represented by a single branch in the collapsed tree. If the probabilities of the branches are assigned to reflect the product $\pi P(e_n,n)$ for each tip node, as shown in Figure 2.6, the expected value $V(e_n)$ will not be altered. The value $V(e_n)$ may be expressed as follows:

$$V(e_n) = \pi P(e_n,n)V(n) + k \tag{2}$$

where k is the expected value of all branches extending outward from e_n except n. Equation (2) simply represents the expected value of node e_n with the single term for node n written separately and the rest expressed as a constant k. Now let V'(n) be a new value for node n such that $V'(n) \geq V(n)$, and such that as a result of assigning V'(n) to n, the value of e_n is raised to equal $V(e_{max})$. Thus,

$$V(e_{max}) = \pi P(e_n,n)V'(n) + k \tag{3}$$

The sensitivity differential of node n is defined to be the difference between V'(n) and V(n). From equation (2) and (3), the difference is given by:

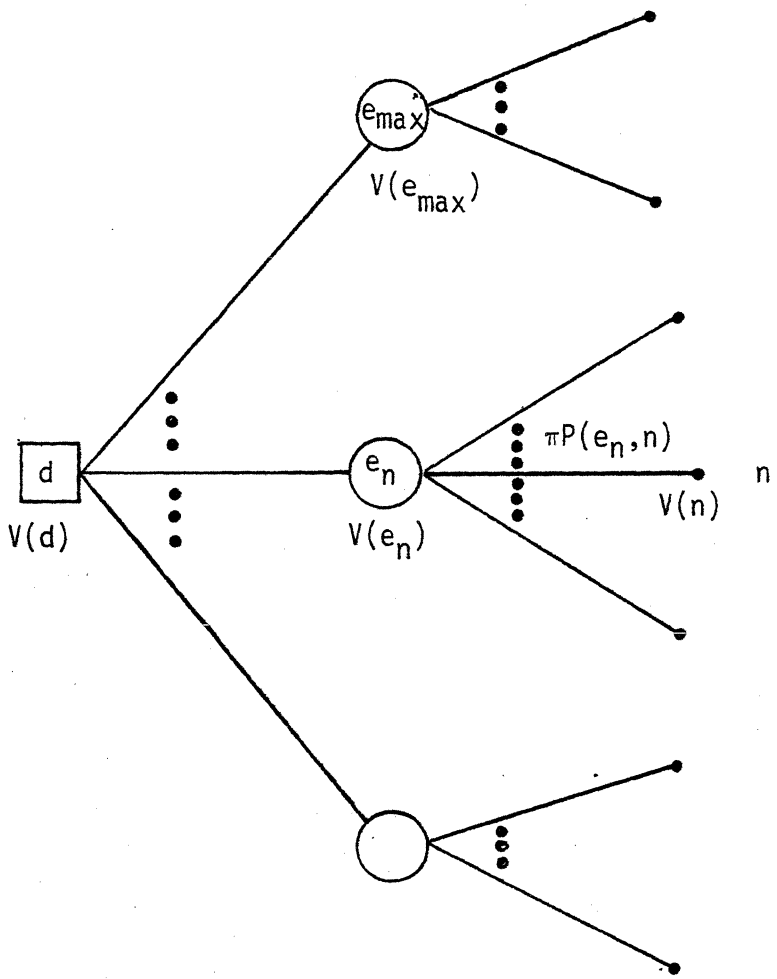


Figure 2.6. Collapsed Event Tree

$$(4) \qquad V'(n)-V(N) = \frac{V(e_{max})-k}{\pi P(e_n,n)} - \frac{V(e_n)-k}{\pi P(e_n,n)}$$

Consequently, from equation (1) we have:

$$(5) \qquad S(n) = \frac{V(e_{max})-V(e_n)}{\pi P(e_n,n)}$$

For notational convenience, this may be written:

$$(6) \qquad S(n) = \frac{\Delta e_n}{\pi P_n}$$

It now remains to show how decision nodes between e_n and n affect the sensitivity differential measure. Figure 2.7 shows a decision tree partitioned into mutually exclusive subtrees such that each subtree has a decision node at its root and contains nothing but event nodes. The sensitivity differential of each tip node may be calculated relative to its own root decision node inside each partitioned subtree. To connect the subtrees, simply consider the value of each subtree as the value of a tip node for the immediately preceeding subtree. That is, every decision node (except the initial one) is seen both as the root of its own subtree and as a tip node of the predecessor subtree.

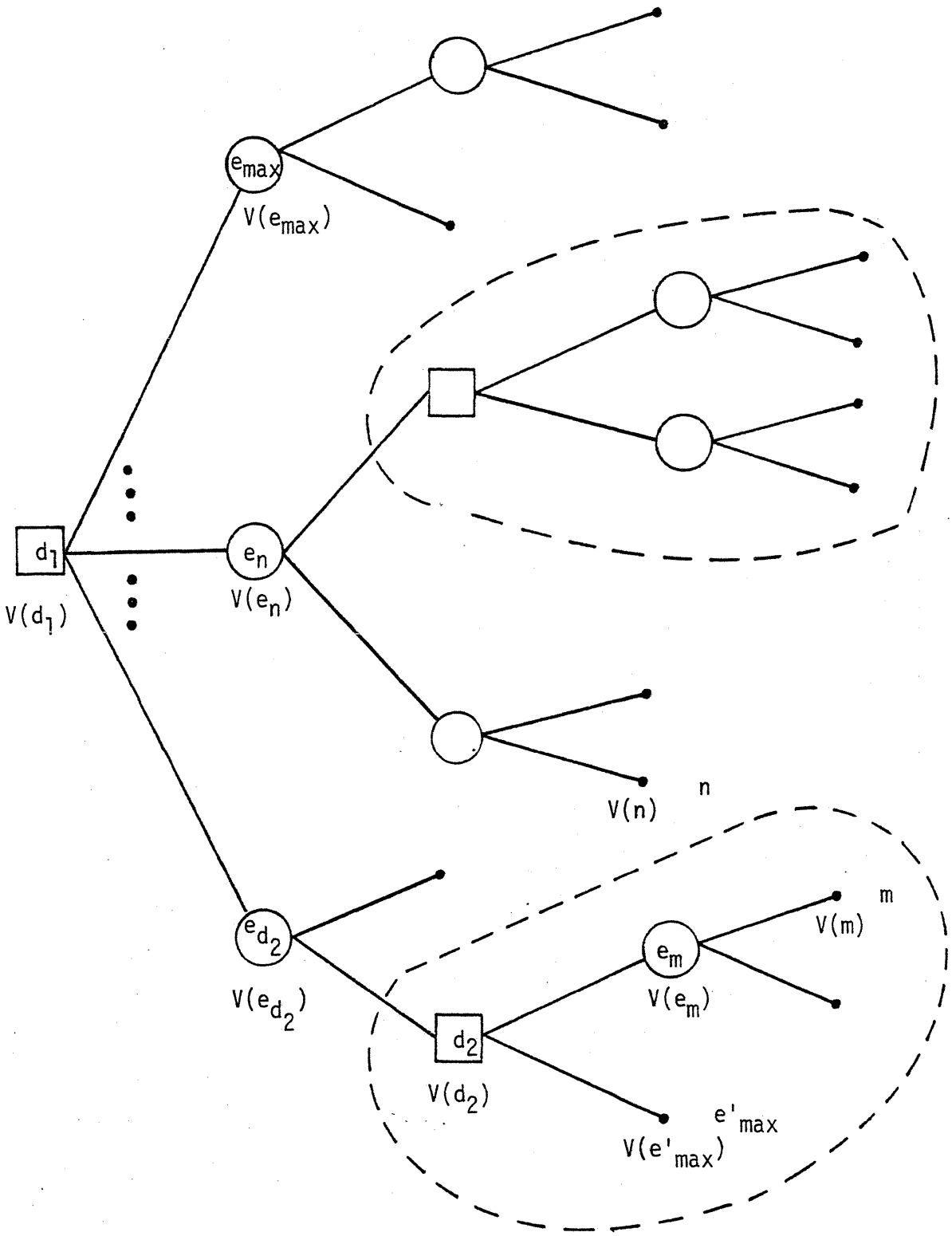


Figure 2.7. Partitioned Decision Tree

In Figure 2.7, node d_1 is the initial decision node of the entire tree and, of course, has as its value the maximum value of its successors. That is,

$$V(d_1) = V(e_{max}) \tag{7}$$

The node d_2 plays two roles. It is the initial decision node of its own subtree and a tip node to the d_1 subtree. Consider node m located in the d_2 subtree with a provisional value of V(m). As shown before in equation (5), it is possible to calculate both the sensitivity differential of node m with respect to d_2 (denoted $S(m/d_2)$) and the sensitivity differential of node d_2 with respect to d_1 (denoted $S(d_2/d_1)$). For reference, these are:

$$S(m/d_2) = \frac{V(e'_{max})-V(e_m)}{\pi P(d_2,m)} \tag{8}$$

$$S(d_2/d_1) = \frac{V(e_{max})-V(e_{d_2})}{\pi P(d_1,d_2)} \tag{9}$$

The task now is to calculate $S(m/d_1)$: the sensitivity differential of node m with respect to d_1.

This is done in a two-step process for the example in Figure 2.7, beginning at d_1 and working forward toward m. Thus,

sensitivity calculations take place in a forward direction (from the root to the tips) as opposed to rollback calculations which take place in the reverse direction (from the tips to the root). The sensitivity differential $S(d_2/d_1)$ is the increment on $V(d_2)$ necessary to bring $V(e_{d_2})$ up equal to the value of $V(e_{max})$. The new, higher value $V'(d_2)$ needed for this is simply the old value plus the differential.

$$V'(d_2) = V(d_2) + S(d_2/d_1) \tag{10}$$

Now, the amount that $V(m)$ needs to be raised to bring $V(d_2)$ up to $V'(d_2)$ is given by:

$$S(m/V'(d_2)) = \frac{V'(d_2)-V(e_m)}{\pi P(d_2,m)} \tag{11}$$

That is, in computing the sensitivity differential of node m, rather than raising $V(e_m)$ up to $V(e'_{max})$, it must be raised *further* to $V'(d_2)$. The value $V'(d_2)$ will always be greater than or equal to that of $V(e'_{max})$ since:

$$V'(d_2) \geq V(d_2) = V(e'_{max}) \tag{12}$$

This two-step process leads to a generalized recursive procedure for finding the sensitivity differential of any node in

the tree with respect to the initial decision node:

$$S(\Gamma(n)) = \begin{cases} \dfrac{S(n)}{P(n)} & \text{for an event node } n \\ S(n)+V(n)-V(\Gamma(n)) & \text{for a decision node } n \end{cases} \tag{13}$$

where $\Gamma(n)$ is the successor of node n and $P(n)$ is the probability along the branch from n to $\Gamma(n)$.

At first glance, the node with the lowest sensitivity differential may appear to be crucial and should be chosen as the next to expand. It may be argued, however, that the factor which determines the node to be selected for expansion is not the absolute sensitivity differential $S(n)$, but the *relative* sensitivity $S_r(n)$:

$$S_r(n) = \frac{\sigma_V(n)}{S(n)} \tag{14}$$

where $\sigma_V(n)$ is the anticipated variation in the provisional value of node n which is likely to take place by further refinements. $\sigma_V(n)$ represents, therefore, the magnitude of the error present in the provisional value estimate $V(n)$ and $S_r(n)$ represents the likelihood that this error would result in a change of plan. The error $\sigma_V(n)$ may depend on the magnitude of the value. For example,

a node with a value of 10 is less likely to be raised to 15 by refinement than a node with a value of 110 is to be raised to 115.

Equation (14) is based on the following noise model. Let v be the provisional value of some tip node as reported during the elicitation interview. Let v* be the "true" value that would result if this node could be theoretically expanded completely. The difference $\Delta v = |v^* - v|$ is the error due to "noise". Figure 2.8 shows these quantities in the form of a probability distribution graph. The value v* is not known. However, treating v* as a random variable, we wish to find the probability that expanding a node with value v will cause a change in the initial decision. Let v_0 be the required value for a decision change, as calculated by sensitivity analysis. Then, we wish to find $P(v^* > v_0 | v)$. To find this probability, some distribution $P(v^* | v)$ must be assumed. We assume here that $P(v^* > v_0 | v)$ is some monotonic increasing function of:

$$\frac{\sigma_v}{v_0 - v} \tag{15}$$

where σ_v^2 is the variance of v*, which leads to equation (14) as a proper basis for node expansion.

The value $\sigma_v(n)$ cannot, of course, be computed exactly. It can, however, be estimated either by asking the decision maker directly to assess the reliability of his value judgment V(n), in the form of a utility interval, or by assuming a reasonable reliability model in the form of a functional relationship $\sigma_v(n) = f(V(n))$

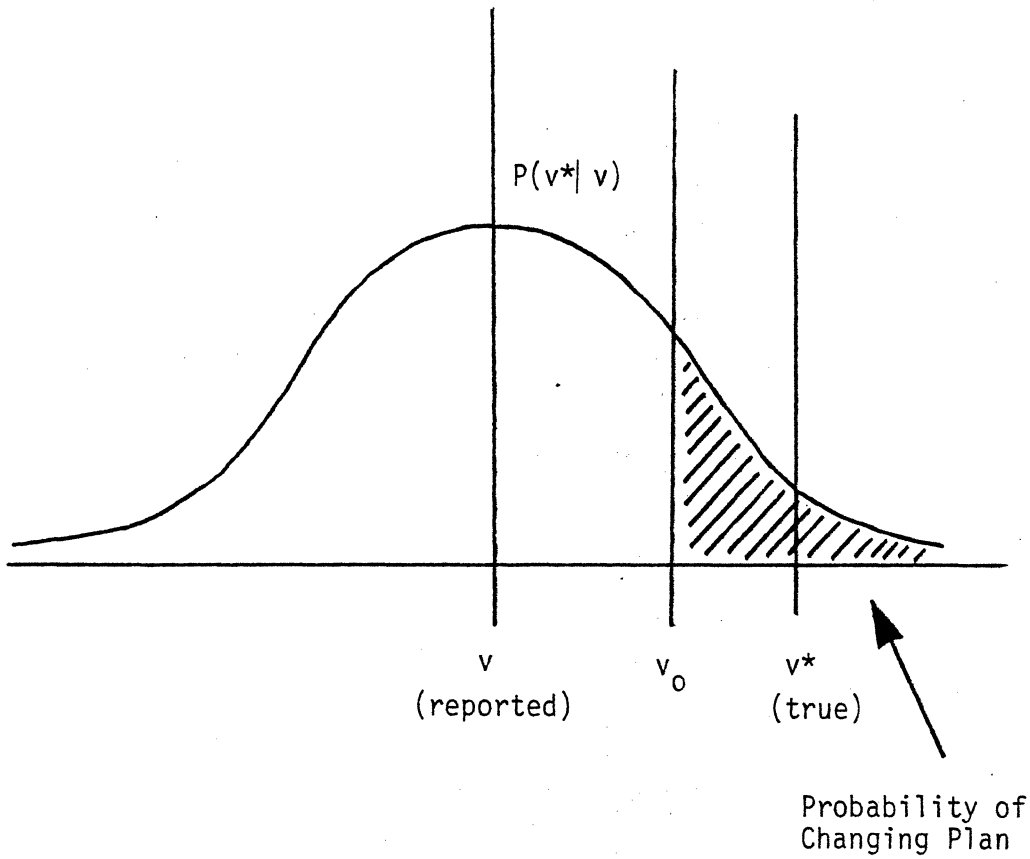


Figure 2.8. Distribution Graph

connecting $\sigma_V(n)$ to the magnitude of the value estimate V(n). The distribution spread may depend on the value of v in a manner shown in Figure 2.9. In the current program version, a linear noise model is assumed: $\sigma_V(n) = a+bV(n)$ reflecting the fact that greater inaccuracies are anticipated in assessing scenarios involving higher stakes. By comparing the provisional value V(n) with its rollback value over many nodes, it is possible to collect statistics on the factors which determine the reliability of human assessments. These could be incorporated to construct more refined models of value reliability for use in subsequent runs.

Once $\sigma_V(n)$ is determined, the value of the relative sensitivity $S_r(n)$ can be computed for all the tip nodes of the partial tree under analysis. The one with the lowest value would be selected for expansion. Needless to say, such analysis cannot be performed during a manual interview, as it involves real-time manipulation of the entire tree at hand.

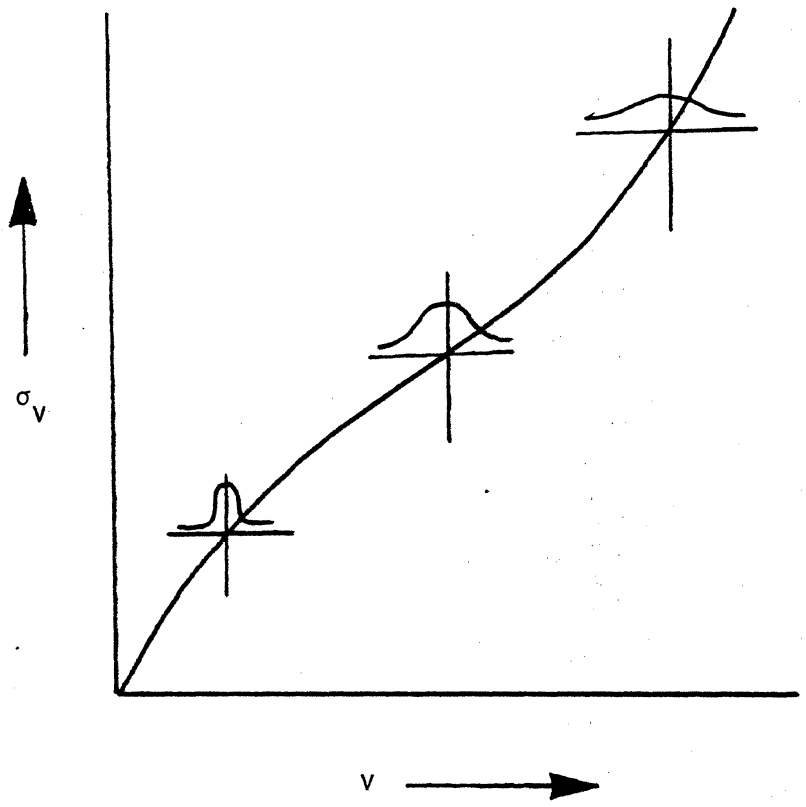


Figure 2.9. σ spread

III. GAINING INFORMATION THROUGH EXPERIMENTATION

Among the more difficult estimates for the decision maker to formulate are the probabilities associated with outcomes of events [11]. Any chance to express and incorporate underlying causes or related events in order to obtain more accurate probability estimates should be exploited. The option to perform an "experiment" that will yield information about the probabilities of a related event provides such a chance. The option may be thought of as "buying information" since there is usually an experimentation cost and part of the analysis is to determine if the information is worth the cost [12]. The experiment may be performed by either an actual physical act (i.e. call your stock broker) or by an internal act of recalling pertinent information (i.e. analyze clues that lead to a certain belief).

Figure 3.1 shows the structure of an experiment. It takes the form of a two-branch decision node followed, on one of the branches, by a single event node. The decision node represents the choice of performing or not performing the experiment. If the experiment is to be performed, the event node represents the possible outcomes or "observations". Each observation has an associated probability of occurrence and the experimentation cost, if there is one, must be incurred before the outcome is known.

Before a successful rollback procedure can take place, all probabilities relevant to the experiment must be determined. They are:

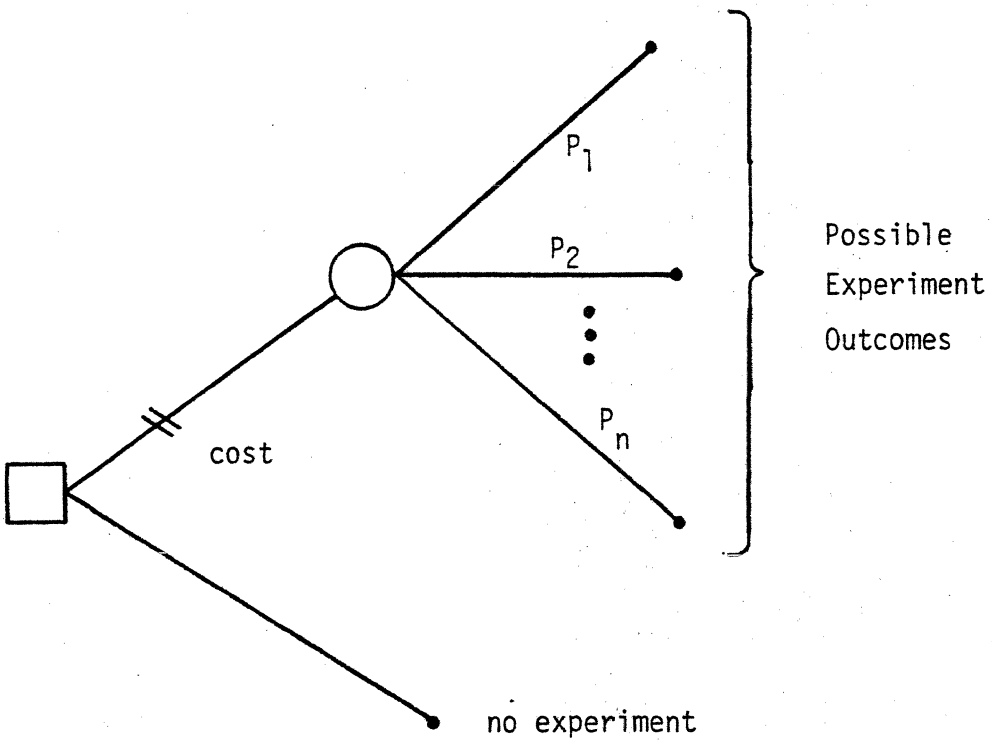


Figure 3.1. Experimentation Structure

$\bigwedge_{j=1}^{m} P(C_j)$ the apriori event probabilities for the outcomes $C_1,..,C_m$,

$\bigwedge_{i=1}^{n} P(E_i)$ the probabilities of the experimental observations $E_1,...,E_n$, and

$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} P(C_j|E_i)$ the observation-conditional probabilities of the event outcomes.

The symbol Λ is used above as a short-hand notation with the following definition:

$$\bigwedge_{i=1}^{n} X_i = X_1,...,X_n$$

The apriori event probabilities $\bigwedge_{j=1}^{m} P(C_j)$ are assumed to be known. However, the probabilities of the experimental observations $\bigwedge_{i=1}^{n} P(E_i)$ are generally not known. The traditional method of obtaining them is by first determining the event-conditional probabilities $\bigwedge_{j=1}^{m} \bigwedge_{i=1}^{n} P(E_i|C_j)$ which are the reverse of the observational-conditional probabilities mentioned above. The $P(E_i)$ are then calculated using probability summing:

$$\bigwedge_{i=1}^{n} P(E_i) = \bigwedge_{i=1}^{n} \sum_{j=1}^{m} P(E_i|C_j)P(C_j)$$

and the observation-conditional probabilities are obtained by using

Bayes' rule:

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} P(C_j|E_i) = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \frac{P(E_i|C_j)P(C_j)}{P(E_i)}$$

As an example of this approach, consider the physician who suspects, on the basis of previously observed clinical signs, that a patient has a specific disease. In order to refine his probability estimation, further tests can be performed that will yield more accurate information about the patient's condition. However, the tests may not be entirely specific to the suspected disease.

The physician may have difficulties estimating the observation-conditional probabilities required, that is, the probability that the patient has the disease given that the test results were positive, etc. The reason is that his knowledge may not be organized in this fashion. It is more likely to be organized around cause-and-effect relationships. It is the disease that causes the positive test results and not the other way around. Thus, the physician is more likely to be able to estimate the event-conditional probabilities: the probability that the test results will be positive given that the patient has the disease, etc. Because of this human tendency to organize knowledge around cause-and-effect relationships, it is more common to elicit the required conditional probabilities in a "reverse" direction, that is, the probability that a particular experiment outcome is observed given that the event "is about to" occur.

If, in fact, the observation-conditional probabilities are easier to elicit than the event-conditional probabilities (as would be the case in purely clinical situations where no disease model exists), then the path probabilities $P(E_i)$ and $P(C_j|E_i)$ can be obtained directly without resorting to Bayes' inversion formula, assuming, of course, that in such situations, the observation probabilities were also directly estimatible.

During the elicitation process, the opportunity for an experiment will normally be discovered at the time probabilities are being estimated for the outcomes to some particular event node. Thus, the experimentation structure must be inserted into a previously generated portion of the decision tree. Figure 3.2 shows the placement of the experiment (indicated by a diamond-shaped node) relative to the affected event node. The experiment must, of course, appear before the event node in the tree but need not be immediately adjacent. It must appear somewhere along the path from the initial root node to the affected event node. This is because the experiment, after all, should be used by the decision maker to change his actions in accordance with its outcomes. There is thus a completely new set of probabilities on the event node for *each* observational outcome of the experiment. These sets are required in addition to the apriori set of probabilities given by the decision maker before the requirement for an experiment arose.

Figure 3.3 shows the traditional representation for an imbedded experiment. Along one of the tree paths extending outward from the primary decision node, the experimentation structure is

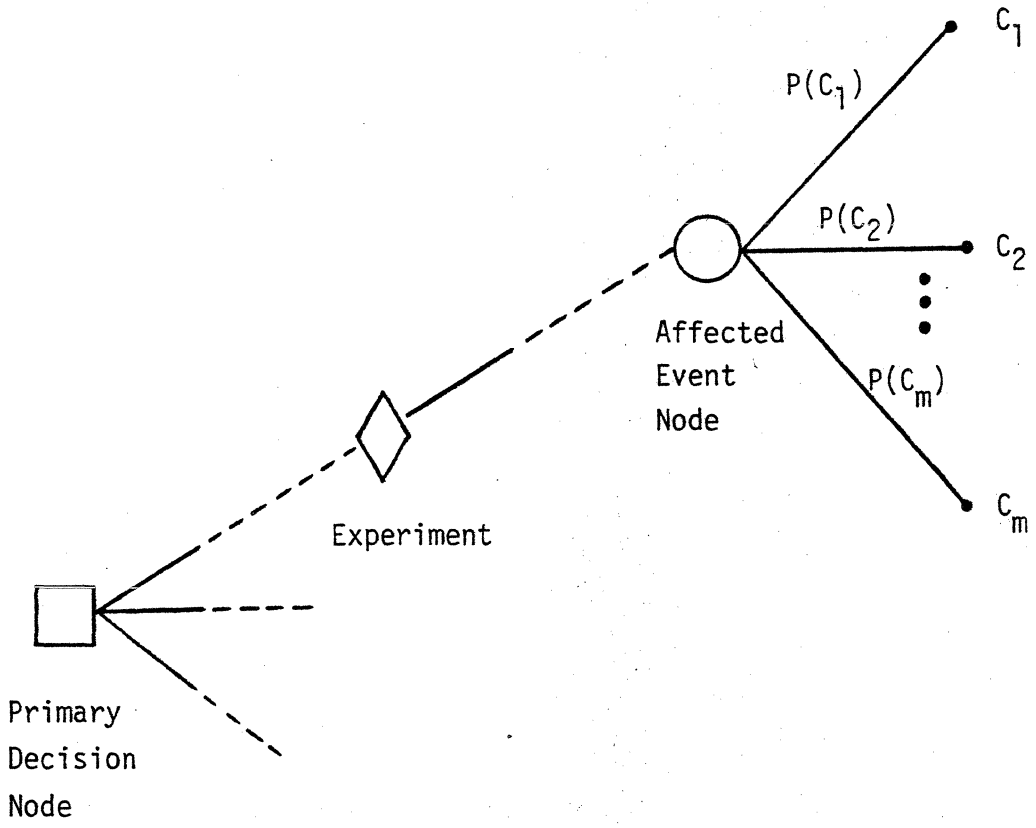


Figure 3.2. Imbedded Experiment

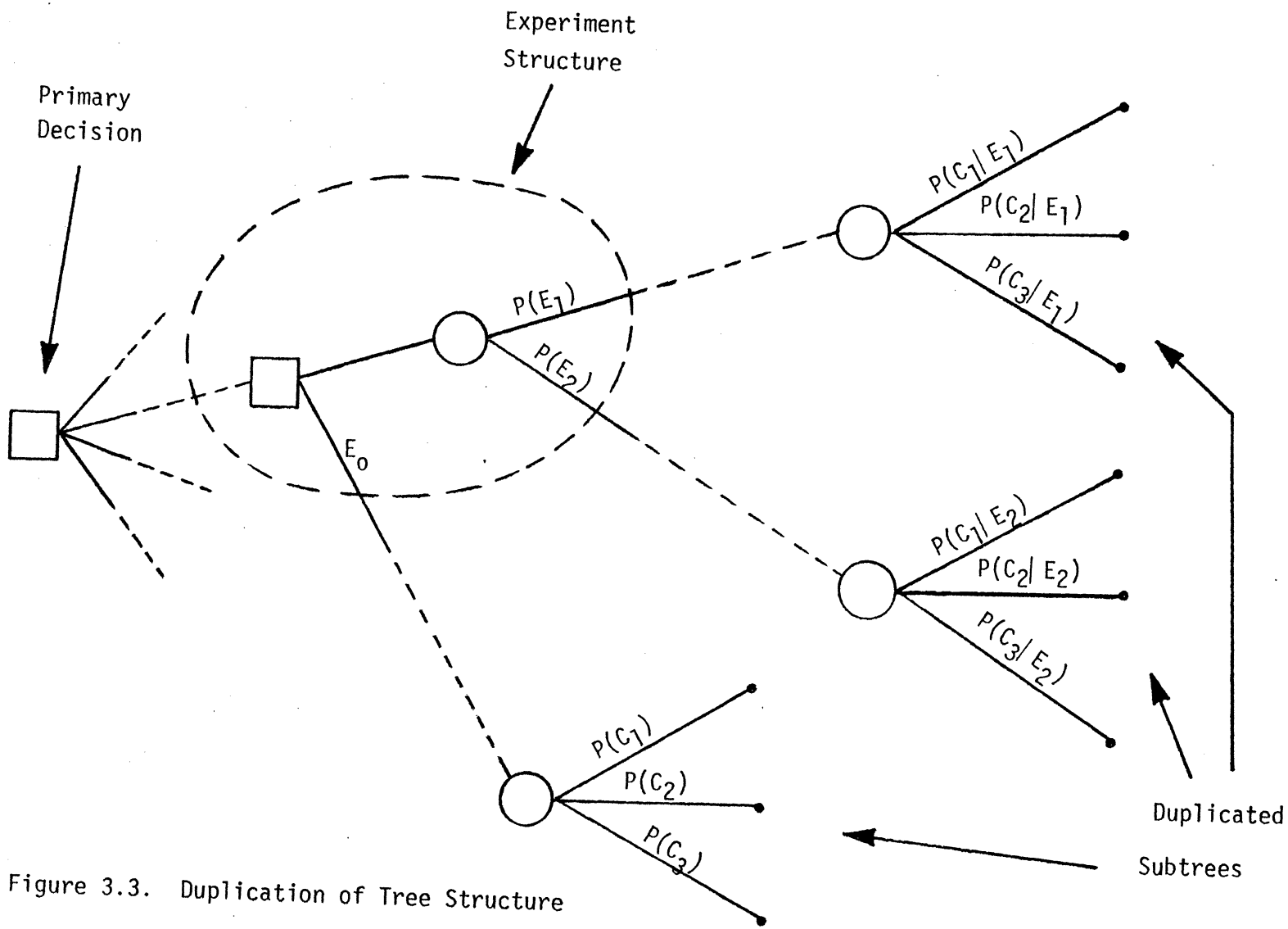


Figure 3.3. Duplication of Tree Structure

encountered. There are n+1 branches extending outward from the structure viewed as a whole. One branch, E_0, leads to the subtree containing the event node with apriori probabilities $P(C_1),\ldots,P(C_m)$ for m branches, where $P(C_j)$ is the apriori probability that event outcome C_j will occur. The other n branches of the experiment lead to n duplicate subtrees, each identical except for the probabilities on the event node. In place of the previously known apriori probabilities, there are mn conditional probabilities $P(C_j|\, E_i)$ for n observational experiment results $i=1,\ldots,n$ and m event outcomes $j=1,\ldots,m$.

Consider, now, the problem of tree duplication as described above (Figure 3.3). In a static environment, subtree duplication caused by the insertion of an experiment structure may not be a hindrance to successful decision analysis. However, in the dynamic environment of interactive tree elicitation, intolerable reference problems arise. Every time a new node is expanded, duplicates must be added to other parts of the tree that have resulted from previous repetitions. Even if a graph structure is allowed, duplication would still occur in the area between the experiment and the affected event node making reference to specific internal nodes difficult.

What is required is a technique that eliminates the necessity for subtree duplication due to an experiment. The solution found to this problem is to consider the entire experimentation structure as consisting only of a *single node with a single branch.* Since the structure is always the same (a two-branch decision node followed by a single event node as shown in Figure 3.1), as long as all relevant

information about the experiment is retained, the problem of subtree duplication disappears.

Figure 3.4 depicts this transformation of the experimentation structure into a single, diamond-shaped node. The cost is placed on the single branch along with the n observation probabilities stored as a vector. The "no experiment" decision branch shown explicitly in the full structure is not shown as part of the experiment node but is, nevertheless, considered to be implicitly part of it.

It is now possible to superimpose all of the duplicated subtrees and form a single unique subtree extending outward from the single experiment node branch. Since the only existing difference in all of the duplicated subtrees is with the conditional probabilities at the affected event node, a single collapsed event node can serve to represent all of the duplicated ones. Figure 3.5 shows the method of storing the conditional probabilities on the branches. Each branch holds a vector of probabilities corresponding to the probability of the particular event outcome given each of the experimental observations respectively. Since each branch contains as many probabilities as experiment outcomes, there are the same number of probabilities in each outcome vector as there are in the single experiment vector (see Figure 3.4). It should be obvious that this representation is isomorphic to the traditional duplicative representation, but forms a much "cleaner" tree.

Having a node in a decision tree with vectors of probabilities on each branch certainly demands a change in the expected-value rollback formulas. The primary objective in the definition of new

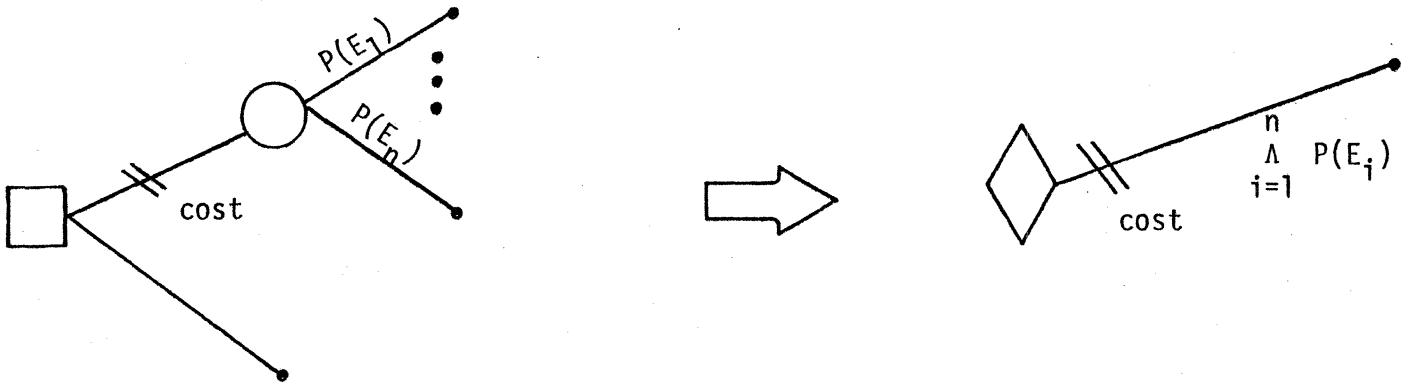


Figure 3.4. Translation to Experiment Node

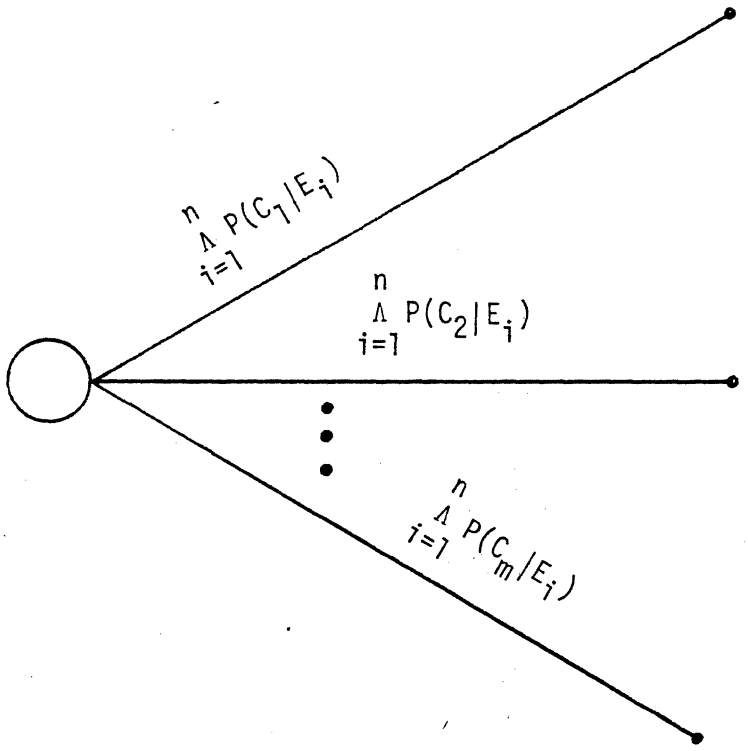
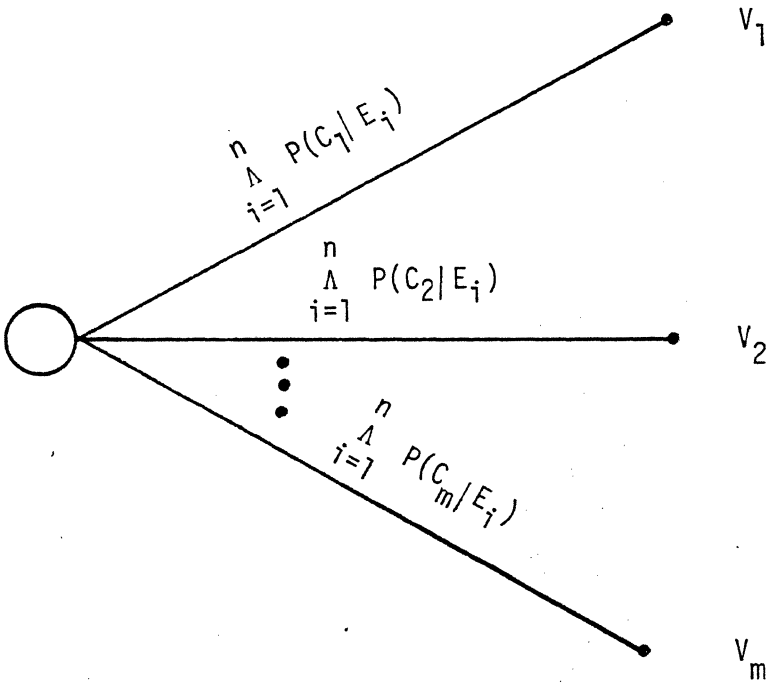


Figure 3.5. Collapsed Event Node

formulas will be, of course, to preserve the isomorphism of representation. That is, the internal values of all of the nodes should be the same regardless of which representation is chosen. Figure 3.6 shows the computation method at the event node containing vectors of observation-conditional probabilities on the branches. Each set of probabilities $P(C_j|E_i)$ is taken, one at a time for i=1,...,n, multiplied by the tip values $V_1, \ldots, V_m$ respectively, and added together. The result is a vector of *values* corresponding to the n values that would have resulted if each event node in the duplicated subtrees had been calculated independently.

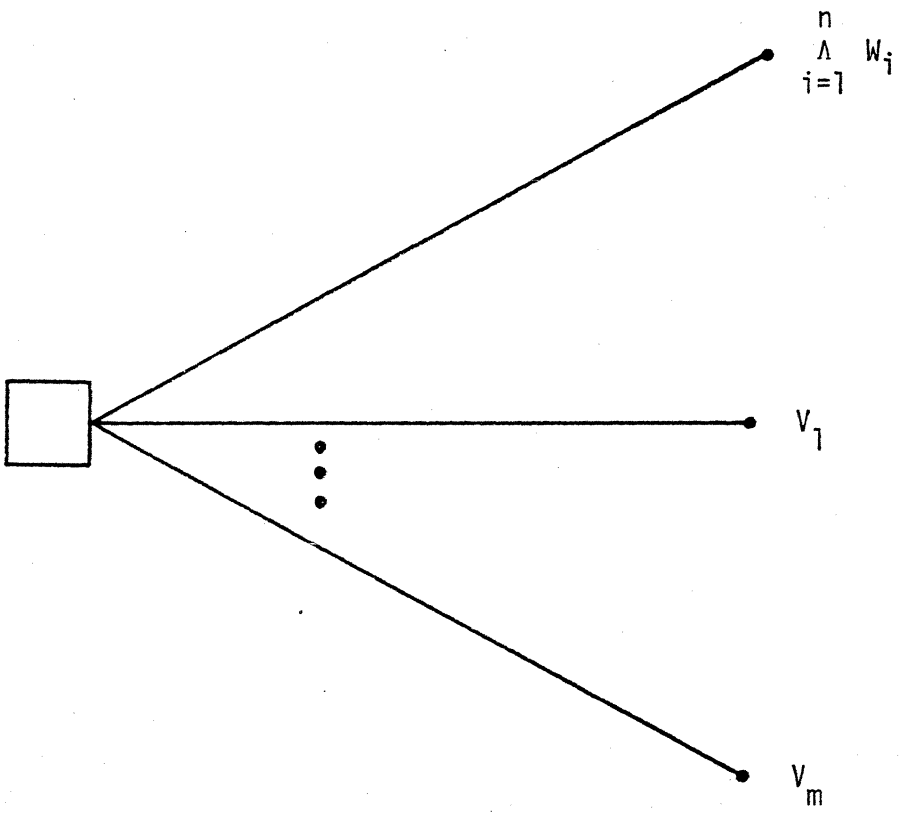
The rollback formula that accepts vectors of probabilities on event nodes is not the only one that must be defined. The predecessor to the node described above will have a vector as one of its tip values. Since this node may be either a decision or an event, two extended rollback formulas must be defined to accept vectors of values at the tips. Figure 3.7 and Figure 3.8 show these two formulas. It is assumed that only one of the branches has a vector and that the others have natural single values. The rollback procedure at the decision node takes the maximum of the combination of all of the single values with each of the vector values respectively (Figure 3.7). The result is, again, a vector of values. Similarly, the event node calculates n expected values using each of the n vector values once, with a vector of n values as the result (Figure 3.8).

Each time the value of a node is calculated, the vector is propagated back toward the root of the tree until the experiment node is reached. Since this node caused the duplication in the first place,



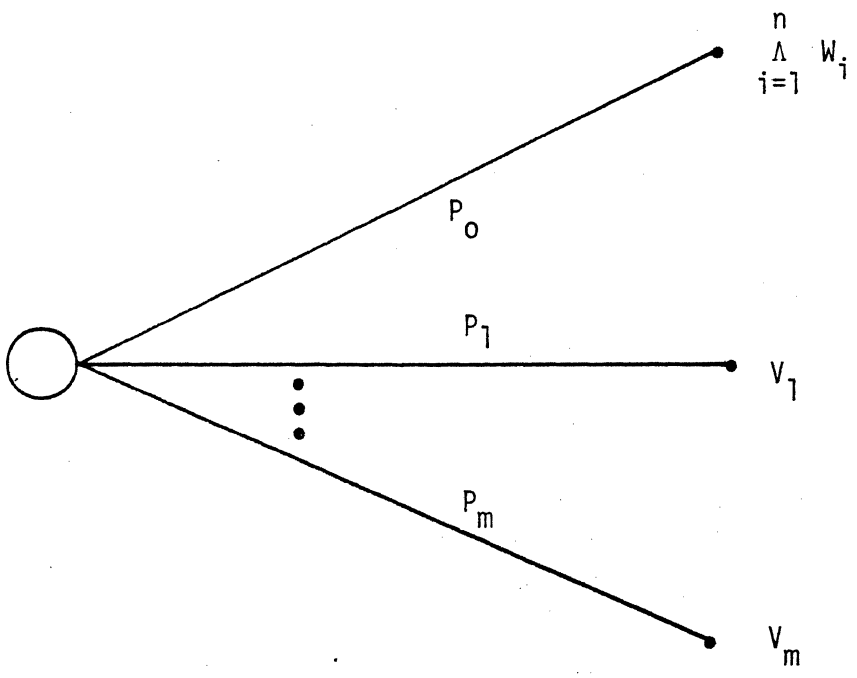
$$\prod_{i=1}^{n} \sum_{j=1}^{m} P(C_j|E_i)V_j$$

Figure 3.6. Event Node Rollback Function with Probability Vectors



$$\bigwedge_{i=1}^{n} \text{MAX}(W_i, V_1, .., V_m)$$

Figure 3.7. Decision Node Rollback Function with a Value Vector

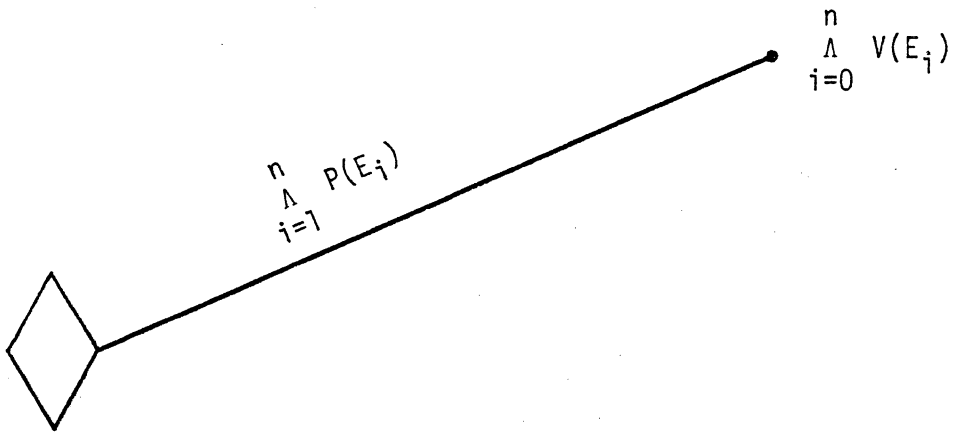


$$\bigwedge_{i=1}^{n} (P_o W_i + \sum_{j=1}^{m} P_j V_j)$$

Figure 3.8. Event Node Rollback Function with a Value Vector

the value of the experiment node is a single number. Figure 3.9 shows how the vector is collapsed. Keeping in mind the "internal structure" of an experiment node (see Figure 3.1), the vector of probabilities on the single branch must have one less member than the vector of values at the tip. This is due to the fact that the experiment structure has one exit branch with no probability attached to it: the "no experiment" option on the decision node. It is assumed, by convention, that the value corresponding to this branch is always contained in the first position of the vector, labelled $V(E_0)$. Thus, it is the task of the rollback function for experiment nodes to take the expected value of the n vector elements i=1,...,n with the n probabilities, respectively, and subsequently compute the maximum of this single value and $V(E_0)$.

It should be clear that if the above procedure is followed, the isomorphism between the two representations will be preserved and the ultimate value of the initial decision node will be the same in either case. The only remaining question is, "What happens when two experiments interact?" It is possible that between one experiment and its corresponding affected event node, another experiment could exist that affects a second event node further down the road. Figure 3.10 shows an example of this situation. Node B is an experiment that helps to estimate the probabilities at node G (shown by the dotted line). Node E is an experiment affecting the probabilities at node H. As might be imagined, the two interacting experiments form a *matrix* of conditional probabilities at node G. In order to clearly represent this situation, Figure 3.11 expands both of the experiment structures in Figure 3.10 and shows all duplications assuming two observations



$$\text{MAX} \left(V(E_0) , \sum_{i=1}^{n} P(E_i)V(E_i) \right)$$

Figure 3.9. Rollback Function for Experiment Nodes

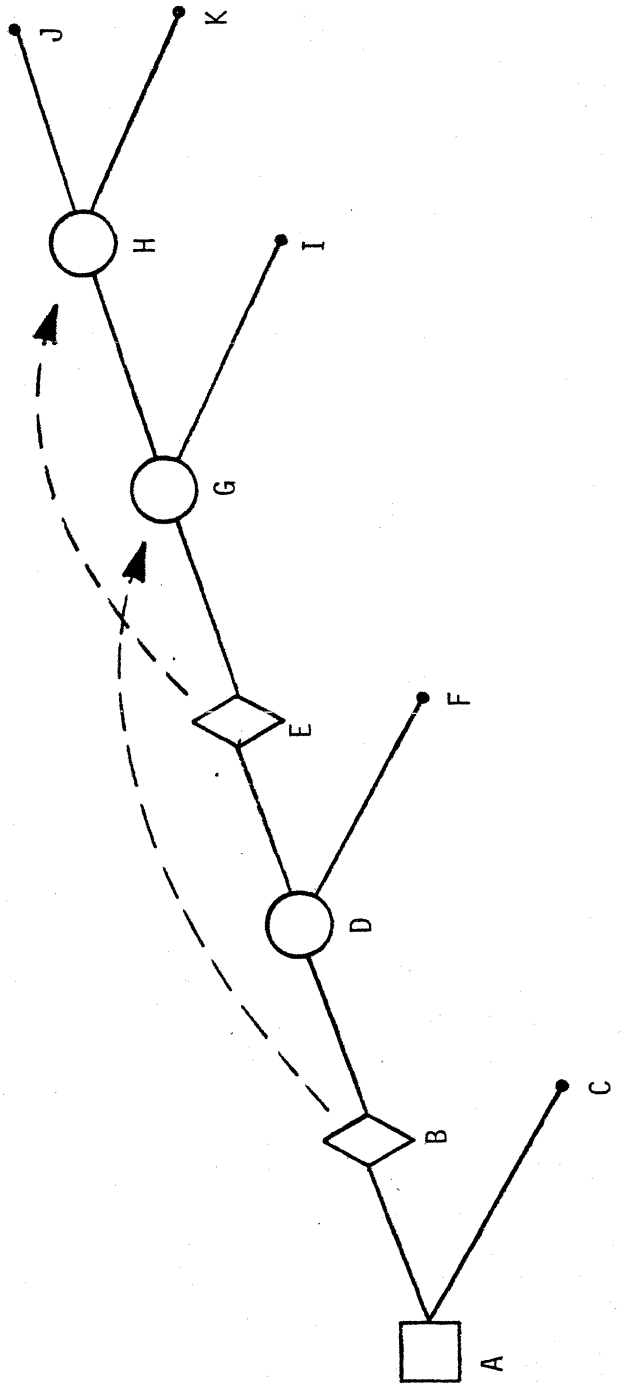


Figure 3.10. Two Intersecting Experiments

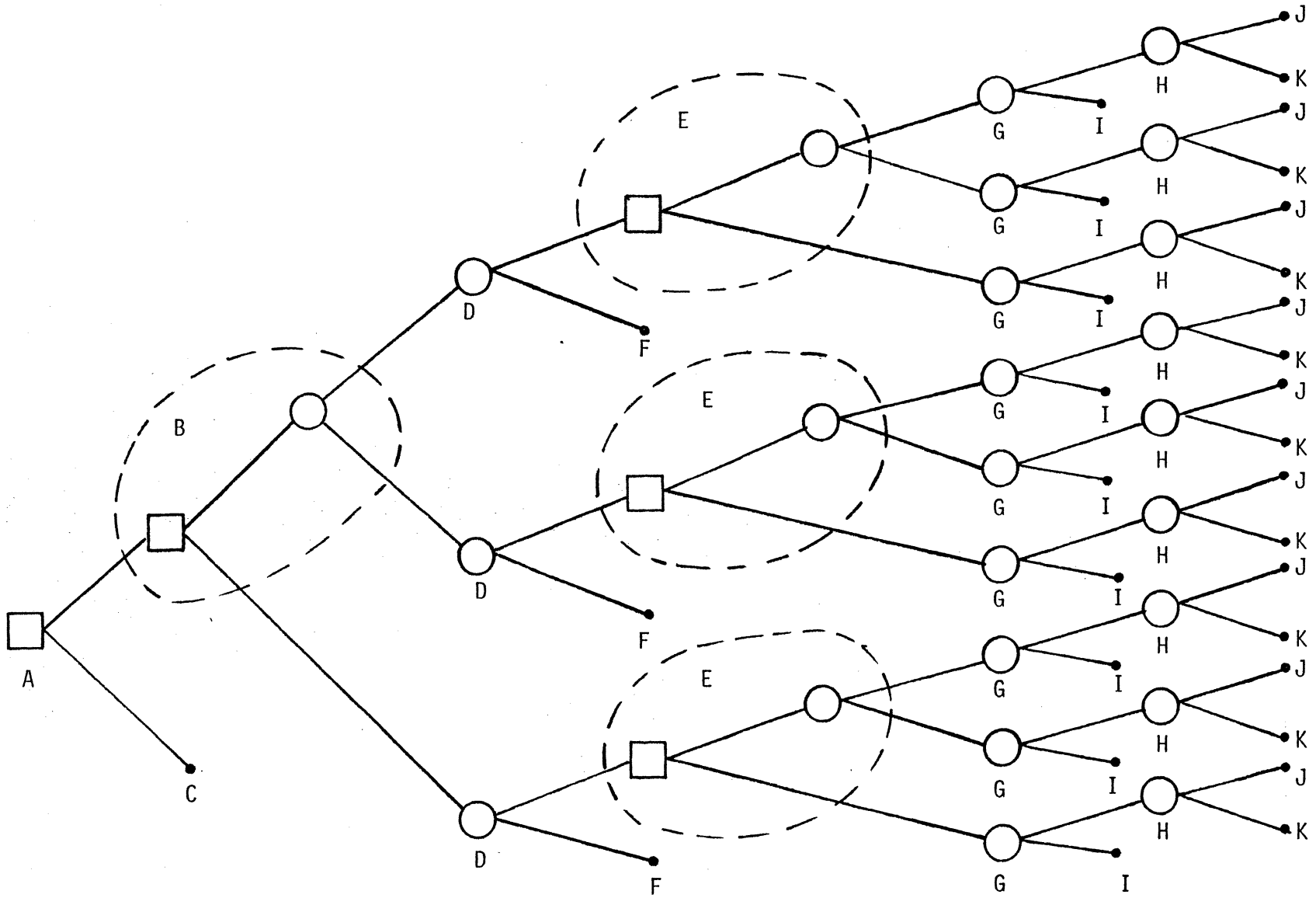
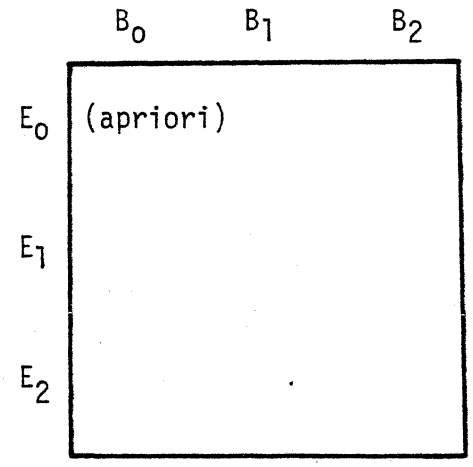


Figure 3.11 Expanded Experiment Structure

for each experiment (three branches in all). The two experiments produce 3^2 duplications of node G and its successors.

Assume that experiment B has observations B_1 and B_2 with a "no experiment" option of B_0. Also, let E_0, E_1, and E_2 stand for these same quantities for the second experiment: node E. Then, the matrix of observation-conditional probabilities would appear as follows.



The apriori probabilities would be found in the (B_0,E_0) entry of the matrix. The other entries would contain all possible combinations of observation-conditional probabilities:

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \bigwedge_{k=1}^{p} P(G_i | B_j, E_k)$$

for each probability $P(G_i)$ extending outward from node G.

Notice the difference in the tree representations in Figure 3.10 and Figure 3.11. Collapsing the experimentation structure into a single node provides a great many advantages with a small burden on computational effort. The tree is cleaner and can be described more compactly with less confusion; events, which are naturally thought of as single entities, are preserved as such; and interactive decision analysis is aided by having a storage representation that is closer to the human conceptual model.

IV. THE ELICITATION PROCEDURE

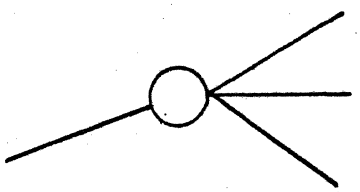
The elicitation procedure is the cycle of logical steps involved in obtaining detailed information from the decision maker about one particular node in the decision tree. The information, which includes provisional values, estimated probabilities, etc., must be sufficient to successfully perform rollback and sensitivity calculations. The information is obtained through a prescribed sequence of interactions and when completed, the node under analysis is said to have been "expanded".

Figure 4.1 shows an overview of the major steps comprising the elicitation procedure and their effect on the expanded node. The first step of the cycle consists of selecting a node for expansion from those available tip nodes. As will be explained later, not all tip nodes are available for further exploration and refinement. The decision maker may have indicated that in certain areas of the tree, enough detail has been established and no further analysis is necessary. Such nodes are called "terminal". The node selection is done through the process of sensitivity analysis described in section II. After node selection is accomplished, the node type must be determined. The type is classed as "decision" or "event" and a list of decision alternatives or event outcomes, as the case may be, is requested. After determining that the members of this "successor" list are mutually exclusive, an iterative procedure is initiated that elicits a provisional value, a probability (if necessary), and a cost for each

1. SELECT NEXT NODE FOR EXPANSION

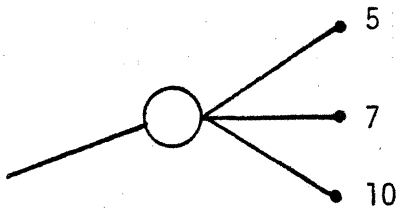
2. DETERMINE NODE TYPE

3. ELICIT ALTERNATIVES OR OUTCOMES

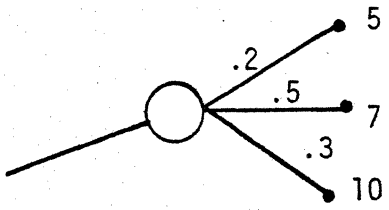


4. DETERMINE IF MUTUALLY EXCLUSIVE

5. ELICIT PROVISIONAL VALUES



6. ELICIT PROVISIONAL PROBABILITIES



7. ELICIT COSTS

8. REQUEST FOR EXPERIMENTATION

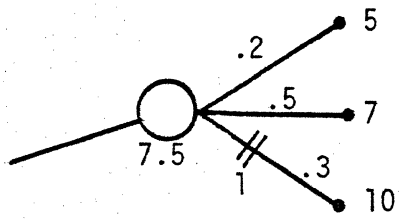


Figure 4.1. Elicitation Procedure

of the members of the list. Then, a request for an experiment is made to ascertain the decision maker's confidence about his estimated probabilities. This finishes the cycle and, after rollback and sensitivity calculations, a new cycle can begin. Each of the above steps will now be described in detail.

Upon the selection of a new tip node to be expanded, the decision maker must be alerted that he is to shift his attention to a specific area of discourse (i.e. a node in the tree). If the node is the successor of a previous decision, typical alert messages would be:

SUPPOSE THAT YOU HAD CHOSEN TO X
ASSUMING THAT "X" WAS PICKED
SUPPOSE THAT "X" IS YOUR CHOICE
WHAT IF YOU CHOOSE TO X
LET US SAY THAT YOU TOOK "X"

The "X" referenced above is the exact name given by the decision maker for the particular decision alternative in question. The alert messages do not form a complete English sentence because they are the precursor to the elicitation request for the node type (see below). If the preselected node is an event rather than a decision, slightly modified alert messages are more appropriate and desirable:

SUPPOSE THAT "X" HAPPENED
WHAT IF "X" OCCURS

ASSUME THAT "X" HAS HAPPENED

LET US SAY THAT IT WAS "X" THAT ACTUALLY TOOK PLACE

where "X" now stands for the event name itself. The alert messages not only serve to orient the decision maker to a particular area of discourse, but also serve as a prelude for interrogation of node type. The messages in the above two sets (as with all message sets following) are phrased to say the same thing in different ways to reduce monotony.

Once the decision maker has been informed as to the area of exploration, the node "type" must be determined. The type will form the basis for classifying the node as a major decision with alternatives or an event with outcomes. The interactive program allows 5 possible node types:

(1) Unknown

(2) Terminal

(3) Decision

(4) Event

(5) Experiment

Before expansion, the node is assigned type "unknown". At expansion time, the node is resolved as being of type "decision", "event", or "terminal". If the node is determined to be of type "terminal", it is assumed that the decision maker is no longer interested in pursuing this particular scenario to any greater level of detail. It is immediately abandoned and its current provisional value (obtained at

a previous time) is considered final. It is removed from the list of available tip nodes for expansion. If the node is not terminal, the decision maker must declare it either decision or event. Finally, nodes of type "experiment" need not be considered here because they are always inserted into previously constructed portions of the tree and never arise as a result of natural node expansion. A detailed discussion of the elicitation of experiment nodes is presented later in this section.

The following queries, which are typical of those used to determine a node of type "decision", are meant to complete the sentence started by the alert messages given above:

IS THERE A DECISION TO BE MADE AT THIS POINT?
WOULD THERE BE OPPORTUNITIES OPEN TO YOU NOW?
ARE SOME OPTIONS AVAILABLE TO YOU?
DO YOU HAVE A CHOICE OF ALTERNATIVES?

If a negative response is received from the above, the following questions probe for "event" nodes:

ARE THERE SOME EVENTS THAT MAY HAPPEN?
ARE EVENTS ABOUT TO HAPPEN OVER WHICH YOU HAVE NO CONTROL?
THEN PERHAPS UNCONTROLLABLE OUTCOMES MAY OCCUR?
CAN YOU THINK OF THINGS THAT MAY HAPPEN AS A RESULT
OF THE CURRENT SITUATION?

Of course, the above questions are asked with the assumption that the responses will have some influence on the decision maker's future courses of action or will be relevant to his problem. If the decision maker fails to give a positive response for either decision or event node types, a terminal node situation is suspected and a confirmation is initiated. It is assumed that enough refinement has occurred on the current scenario and that the decision maker is ready to explore other areas in more depth. Typical confirmation queries are:

DO YOU WISH TO STOP EXPLORING FURTHER IN THIS DIRECTION?
HAS THERE BEEN ENOUGH DETAIL EXPRESSED SO FAR?
SHOULD WE EXPLORE SOME OTHER POSSIBILITIES IN ANOTHER AREA?
PERHAPS WE SHOULD TALK ABOUT SOMETHING ELSE?
I ASSUME THAT WE CAN LEAVE THIS SITUATION?

If a negative response is received to the above questions, it is assumed that some misunderstanding has taken place and a short tutorial is started with the intention of ascertaining which of the three node types is preferable and forcing a selection among them. If the decision maker indicates two terminal nodes in succession, he is asked if he wishes to end the interview.

With the node type determined, elicitation of decision alternatives or event outcomes can commence. The decision maker is simply asked to list them using short descriptive phrases. There is no limit to the length of the list but it must contain at least two alternatives. Typical request messages for decision alternatives are:

PLEASE LIST THE ALTERNATIVES THAT YOU HAVE, ONE AT A TIME.

STATE THE CHOICES THAT YOU HAVE.

LIST THE OPTIONS THAT ARE AVAILABLE.

PLEASE LIST THE DECISIONS THAT YOU COULD MAKE.

TELL ME WHAT IS AVAILABLE.

WHAT OPPORTUNITIES ARE THERE?

A simple keyword search on the response determines when the list is complete. Typical requests for outcomes to events are worded as follows:

PLEASE LIST THE OUTCOMES.

EXACTLY WHAT EVENTS COULD OCCUR?

STATE THOSE EVENTS THAT MAY HAPPEN.

LIST THE POSSIBLE OCCURRENCES THAT YOU FORESEE.

WHAT DO YOU SUPPOSE COULD HAPPEN?

With each response of a decision alternative or an event outcome as the case may be, a new successor node skeleton is constructed and storage allocated to accommodate further information as it is obtained.

Because of the strict requirement for mutually exclusive decision alternatives and event outcomes, a confirmation must be received from the decision maker that the successors just listed are

indeed mutually exclusive. For decision nodes, the following questions confirm this fact:

ARE THE ALTERNATIVES MUTUALLY EXCLUSIVE?

DOES THE CHOICE OF ONE ALTERNATIVE EXCLUDE THE OTHERS?

IS ONLY ONE CHOICE POSSIBLE?

IS IT TRUE THAT CHOOSING ONE OF THESE ALTERNATIVES EXCLUDES THE OTHERS FROM BEING CHOSEN?

AM I CORRECT IN ASSUMING THAT ONLY ONE CHOICE IS POSSIBLE?

For event nodes, the following queries are typical:

ARE THESE EVENTS MUTUALLY EXCLUSIVE?

DOES THE OCCURRENCE OF ONE EVENT EXCLUDE THE OTHERS FROM HAPPENING?

CAN JUST ONE OUTCOME HAPPEN AT A TIME?

AM I CORRECT IN ASSUMING THAT ONLY ONE CAN OCCUR?

At this stage in the elicitation process, the preselected node has been completely expanded in the sense that all relevant information has been obtained about it in particular. It is now necessary to acquire enough information about the node's successors so that adequate rollback and sensitivity analysis can be performed in preparation for the selection of another node to expand. The successors, as listed by the decision maker, are individually considered by introducing them one at a time with short alert messages

followed by requests for provisional value, probability (if necessary), and cost. By iterating on each successor in this way, the decision maker is not forced to jump from one successor to another while trying to estimate values and probabilities, etc. The short alert messages are:

LET'S CONSIDER "X" FOR A MOMENT.
NOW CONSIDER "X".
WHAT ABOUT "X"?
LET'S TALK ABOUT "X" FOR A WHILE.

Once the decision maker has been informed about the area of consideration, the first quantity to be requested is the provisional node value, that is, the worth to him of the possible opportunities that this particular situation may open. This value may be given in absolute terms, such as money, or in a relative utility scale that indicates personal preference. The only criteria is consistency. Typical requests for provisional node value are:

TRY TO PLACE A NUMERIC VALUE ON THIS SITUATION.
WHAT VALUE WOULD YOU GIVE TO THIS OPPORTUNITY?
HOW WOULD YOU EVALUATE THIS SITUATION?
ESTIMATE THE VALUE IF YOU WERE IN THIS POSITION.
WHAT IS THE POSSIBILITY WORTH TO YOU?

If, by previous questioning, the selected node was determined to be an event, probabilities must be assigned to each of the successor

nodes. It is assumed that the decision maker is capable of providing rough probability estimates without a complicated elicitation process. Typical queries for probability are as follows:

WHAT IS THE PROBABILITY OF THIS OUTCOME?
WHAT ARE THE CHANCES THAT THIS EVENT WILL OCCUR?
HOW MUCH OF A CHANCE DOES IT HAVE?
WHAT PROBABILITY WOULD YOU PLACE ON IT?

Of course, no probability questioning occurs on the successors of decision nodes. Further, since the sum of the probabilities on all successors must equal 1, no questioning is necessary on the last successor.

The final item of information that must be obtained for each of the successors is the cost. If the decision maker has indicated, through previous questioning, that he wishes to use a utility scale of his own choosing in order to express provisional values, it is assumed that any incurred costs will be absorbed in the utility estimate. However, if he uses an absolute scale (i.e. money [13]) for representing provisional values, the cost of each decision alternative or event outcome is required. This cost is always assumed to be local and independent of the provisional value. That is, the cost will remain constant even though the provisional value may change due to further refinement. When calculating rollback, the cost is incorporated

into the standard formulas as follows:

$$V(N) = \underset{i}{\text{MAX}} \, [V(N_i)-C(N_i)]$$

where N is a decision node, N_i are the successors of N, and C is the cost function. For event nodes, using the above notation:

$$V(N) = \sum_i (P(N_i)[V(N_i)-C(N_i)])$$

The following queries are typical of those used to elicit cost:

WHAT IMMEDIATE COST IS EXPECTED?
WHAT WOULD BE THE IMMEDIATE COST, ASSUMING THIS SITUATION?
STATE THE COST IF THERE IS ONE.
IF THERE IS AN ASSOCIATED COST, WHAT IS IT?
WHAT IMMEDIATE COST IS ANTICIPATED?

With the provisional values, probabilities, and costs obtained for each successor, the elicitation procedure is completed except for a possible option to perform an experiment. As described in detail in section III, the purpose of an experiment is to aid in estimating the probabilities of an event node. Of course, if the previous elicitation cycle was engaged in the analysis of a decision node, no request for an experiment is made. The following are typical questions that request for an experiment option:

CAN YOU IMPROVE THE PROBABILITY ASSIGNMENTS BY PERFORMING
AN "EXPERIMENT"?
ARE YOU UNHAPPY WITH THE ACCURACY OF THESE PROBABILITY
ESTIMATES?
IS IT POSSIBLE TO DO ANYTHING IN ORDER TO IMPROVE THE
ABOVE PROBABILITY VALUES?

If the option is declined, a new elicitation cycle begins by selecting a new node for expansion. If it is accepted, the experiment elicitation process is initiated that begins with a request for the experiment name and information concerning the time at which it is to be performed.

WHAT TYPE OF EXPERIMENT COULD BE PERFORMED?
WHEN WOULD IT BE CONDUCTED?

The decision maker is expected to respond with the name of a previously expanded internal node. The program searches the path from the root to the current tip under analysis and selects a node that is the closest match to the received response. Thus, it is not necessary for the decision maker to remember the exact name given to each node. The tree is broken at this point with storage allocated to accomodate the new node. The next step is to ascertain the possible experiment outcomes:

PLEASE LIST THE POSSIBLE OBSERVATIONAL OUTCOMES
OF THE EXPERIMENT.

The final step in the experiment elicitation cycle is to obtain the event-conditional probabilities at the affected event node just expanded. With m event outcomes and n experiment observations, there are mn conditional probabilities. By using the fact that n sets of these probabilities must add to 1, the total is reduced to n(m-1) elicited probabilities. Each event outcome is taken in turn with each of the experiment outcomes as follows:

SUPPOSE THAT "C_j" IS ABOUT TO HAPPEN...
WHAT IS THE PROBABILITY OF "E_1" HAPPENING?
WHAT IS THE PROBABILITY OF "E_2" HAPPENING?
...etc.

where C_j is the jth event outcome and E_1, E_2, etc. are the experiment observations.

The request for experimentation completes the elicitation procedure for the expansion of a single node. The cycle of rollback, sensitivity analysis, node selection, and node expansion is repeated again and again until either all tip nodes have been removed from a status of "available for expansion" or the decision maker has requested to prematurely end the interview. At this time, the tree is closed and, after a final rollback calculation, he is made aware of the most promising scenario discovered.

The summary information concerning the results of the interview is more than a simple indication of the best recommendation for the initial choice of alternatives. The decision maker is given a complete

"plan of action" that will tell him what decision to make at every decision node in the optimal solution subtree. This is that subtree which, starting from the root, takes the maximum-valued branch at each decision node and *all* branches at each event node. In addition, experiment nodes are given special consideration. Each decision that comes under the influence of the experiment is mentioned separately with recommended actions for each observational outcome so that the decision maker may obtain the most information from the experiment. The following flowchart (Figure 4.2) diagrams the complete elicitation procedure in a compact form and describes its overall logical structure.

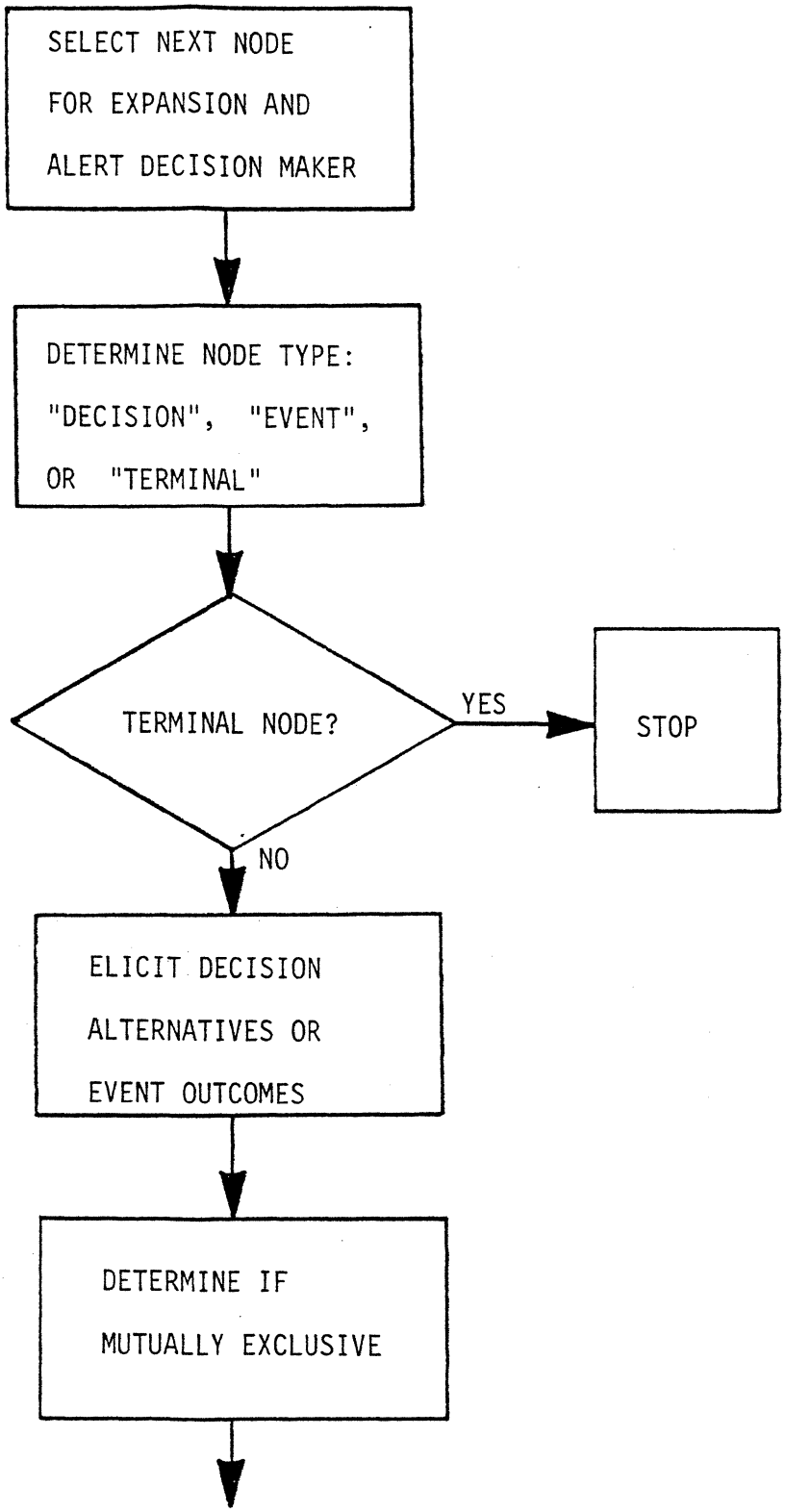


Figure 4.2. Elicitation Procedure Flowchart

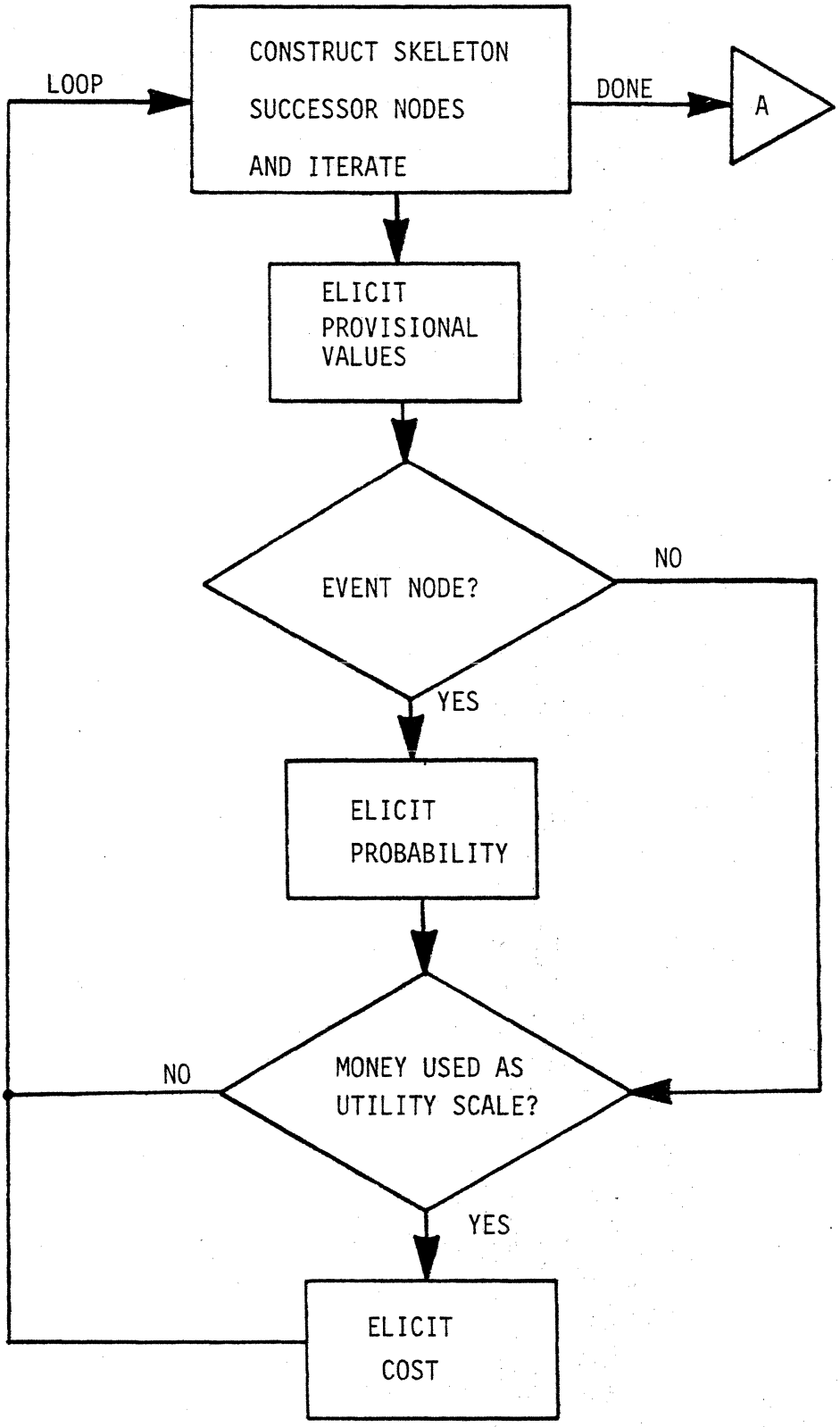


Figure 4.2. Elicitation Procedure Flowchart (Continued)

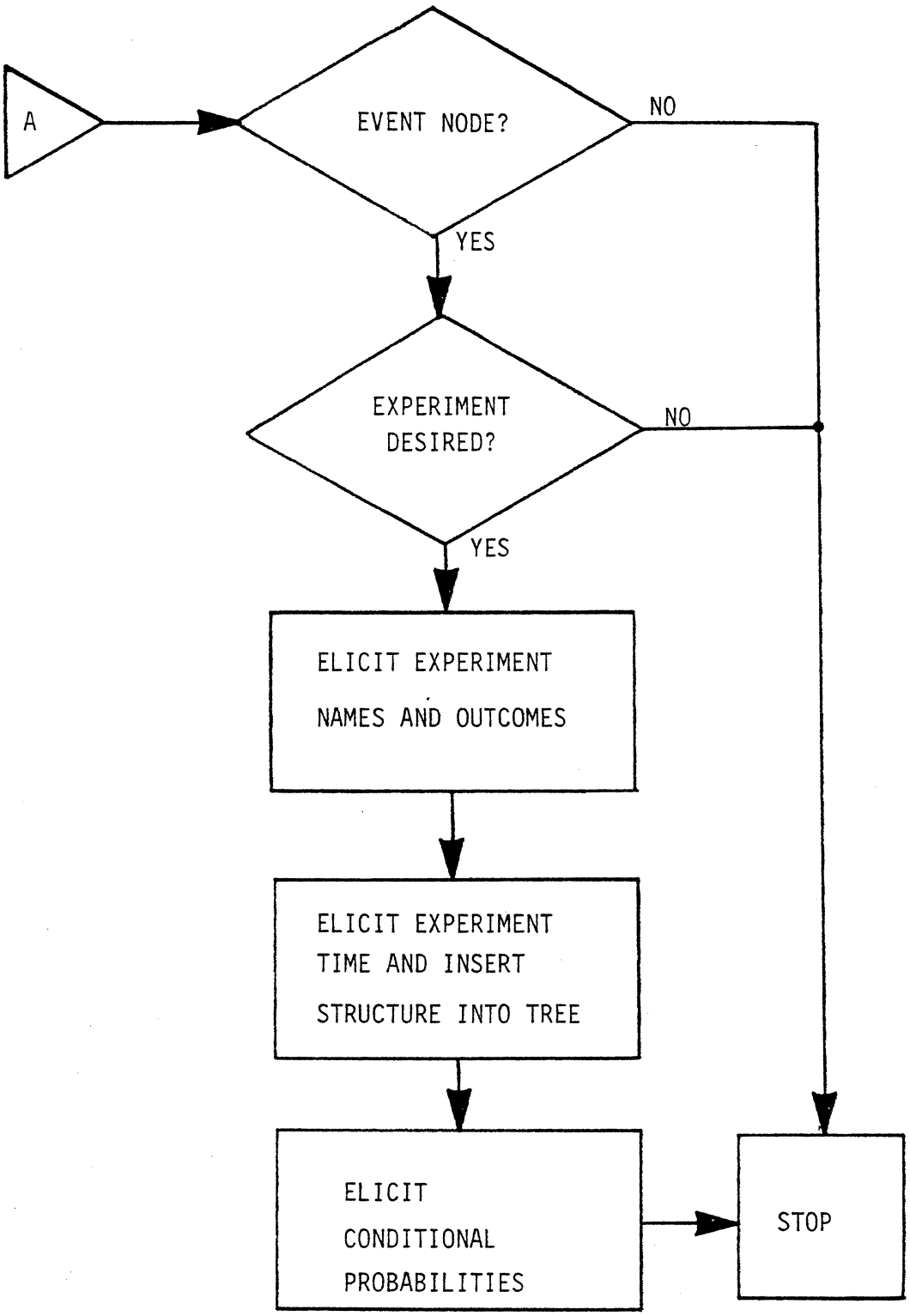


Figure 4.2. Elicitation Procedure Flowchart (Continued)

V. SAMPLE INTERVIEW SESSION

The following hypothetical situation was chosen as an example for demonstrating the program operation. We imagine a scientist who is facing the dilemma of sending his brilliant proposal either to governmental agency X or to agency Y. (The example is not altogether unrealistic from the environment surrounding the funding of *this* research.) It is assumed, because of inter-agency code of conduct, that he cannot send the proposal to both agencies simultaneously. Agency Y has already indicated interest in his work and a willingness to support it at a level of $50,000. Agency X, on the other hand, has not yet had an opportunity to appraise it and further, is not committing funds until the end of the year (i.e. two months hence). The potentially available funds from agency X are much higher: full funding at a level of $100,000 or partial funding at $70,000 (see Figure 5.1).

The basic decision that the scientist must make at the moment is whether to send the proposal to agency Y immediately or to hold it for two months for the purpose of improving it to fit some of the specific needs of agency X. After the two months has elapsed, the opportunity to send it to agency Y is still open. However, some erosion in certainty would result due to the delay in submission.

During the elicitation of the probability estimates concerning the funding from agency X, the scientist feels uncertain about the values he reports. He realizes that the delay in submission will also offer him the opportunity to gain information by soliciting

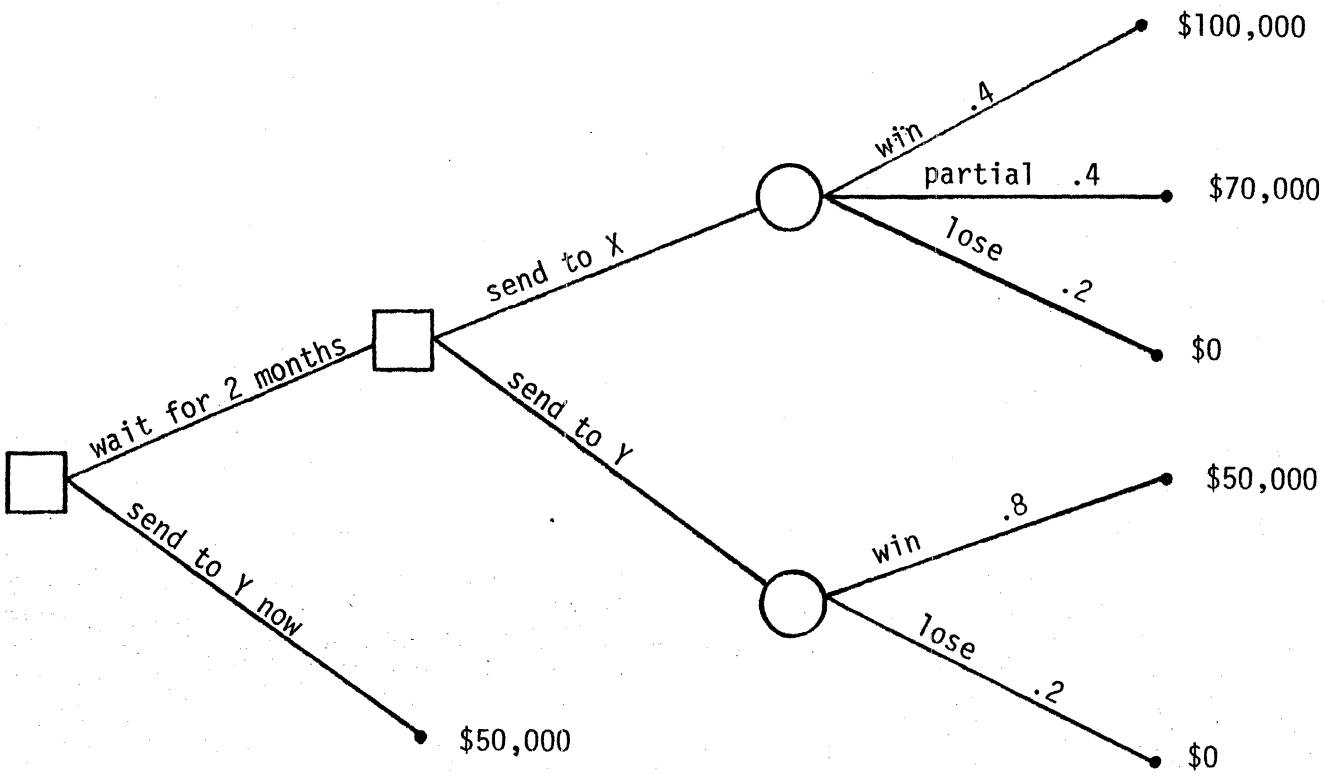


Figure 5.1. The Basic Decision Structure

the opinion of an official from agency X as to the technical compatibility of the proposal with respect to the overall agency objectives. The official cannot be expected to give more than a "favorable" or "adverse" response. Through past experience, the scientist is aware of the official's tendency to respond over-optimistically even regarding proposals which are not about to be funded. This tendency can be formulated in terms of the event-conditional probabilities of the experiment associated with evaluating the proposal.

The following interview session demonstration is reproduced directly from the computer output. An asterisk is the signal from the computer that it is ready to receive input from the user. Thus, text immediately following an asterisk is considered to have been typed by the scientist. Following the interview are two figures showing the resulting decision structure. Figure 5.2 shows all of the duplicated subtrees due to the experiment and Figure 5.3 shows the condensed structure using experiment nodes.

WHAT'S YOUR PROBLEM? *SENDING THE PROPOSAL TO AGENCY Y OR WAITING

PLEASE LIST THE ALTERNATIVES THAT YOU HAVE, ONE AT A TIME.

1.*SUBMIT TO AGENCY Y IMMEDIATELY

2.*WAIT FOR TWO MONTHS

3.*NO MORE

ARE THE ALTERNATIVES MUTUALLY EXCLUSIVE?

*YES

NOW CONSIDER "SUBMIT TO AGENCY Y IMMEDIATELY".

ASSUMING THAT THE CURRENT SITUATION WERE TRUE, TRY TO ESTIMATE HOW GOOD IT IS INDEPENDENTLY OF OTHER POSSIBILITIES. YOU MAY USE MONEY AS A SCALE OR ONE OF YOUR OWN CHOOSING, HOWEVER, YOU MUST BE CONSISTENT AND USE THE SAME SCALE THROUGHOUT THE INTERVIEW.

DO YOU WISH TO USE MONEY AS A SCALE?

*YES

ENTER YOUR ESTIMATE.

*50000

ARE THERE ANY HIDDEN COSTS NOT TAKEN INTO ACCOUNT IN THE PREVIOUS VALUE ESTIMATION?

*NO

LET'S LOOK AT "WAIT FOR TWO MONTHS"

WHAT VALUE WOULD YOU GIVE TO THIS SITUATION?

*70000

WHAT IMMEDIATE COST IS EXPECTED?

*0

SUPPOSE THAT YOU HAD CHOSEN TO WAIT FOR TWO MONTHS

CAN YOU THINK OF THINGS THAT MAY HAPPEN AS A RESULT?

*NO

IS THERE A DECISION TO BE MADE AT THIS POINT?

*YES

STATE THE CHOICES THAT YOU HAVE.

1.*SEND TO X

2.*SEND TO Y

3.*DONE

DOES THE CHOICE OF ONE ALTERNATIVE EXCLUDE THE OTHERS?

*YES

NOW CONSIDER "SEND TO X"

TRY TO PLACE A NUMERIC VALUE ON THIS SITUATION.

*70000

WHAT WOULD BE THE IMMEDIATE COST, ASSUMING THIS SITUATION?

*0

LET'S LOOK AT "SEND TO Y"

HOW WOULD YOU EVALUATE THIS SITUATION?

*50000

HOW MUCH EXPENSE IS ANTICIPATED?

*0

ASSUMING THAT "SUBMIT TO AGENCY Y IMMEDIATELY" WAS PICKED,

ARE EVENTS ABOUT TO HAPPEN OVER WHICH YOU HAVE NO CONTROL?

*NO

DO YOU HAVE A CHOICE OF ALTERNATIVES?

*NO

DO YOU WISH TO STOP EXPLORING FURTHER IN THIS DIRECTION?

*OK

SUPPOSE THAT "SEND TO Y" IS YOUR CHOICE.

ARE THERE SOME EVENTS THAT MAY HAPPEN?

*YES

PLEASE LIST THE OUTCOMES.

1.*ACQUIRE FUNDING FROM Y

2.*Y REJECTS PROPOSAL

3.*DONE

ARE THESE EVENTS MUTUALLY EXCLUSIVE?

*YES

NOW CONSIDER "ACQUIRE FUNDING FROM Y"

ESTIMATE THE VALUE IF YOU WERE IN THIS POSITION.

*50000

TRY TO ESTIMATE THE PROBABILITY THAT THIS EVENT WILL HAPPEN.

*0.8

IF THERE IS AN ASSOCIATED COST, WHAT IS IT?

*0

LET'S LOOK AY "Y REJECTS PROPOSAL"

WHAT IS THE SITUATION WORTH TO YOU?

*0

STATE THE IMMEDIATE COST IF THERE IS ONE.

*0

CAN YOU IMPROVE THESE PROBABILITY ASSIGNMENTS BY

PERFORMING AN EXPERIMENT?

*NO

SUPPOSE THAT "ACQUIRE FUNDING FROM Y" HAPPENED.

ARE THERE SOME OPTIONS AVAILABLE TO YOU?

*NO

COULD UNCONTROLLABLE OUTCOMES OCCUR?

*NO

HAS THERE BEEN ENOUGH DETAIL EXPRESSED SO FAR?

*YES

WHAT IF "Y REJECTS PROPOSAL" OCCURS?

WOULD THERE BE OPPORTUNITIES OPEN TO YOU NOW?

*NO

CAN YOU THINK OF THINGS THAT MAY HAPPEN AS A RESULT?

*NO

SHALL WE TERMINATE THE INTERVIEW?

*NO

WHAT IF YOU CHOOSE TO SEND TO X?

ARE EVENTS ABOUT TO HAPPEN OVER WHICH YOU HAVE NO CONTROL?

*YES

EXACTLY WHAT EVENTS COULD OCCUR?

1.*FULL FUNDING FROM X

2.*PARTIAL FUNDING FROM X

3.*X REJECTS PROPOSAL

4.*NO MORE

DOES THE OCCURRENCE OF ONE EVENT EXCLUDE THE OTHERS FROM HAPPENING?

*YES

NOW CONSIDER "FULL FUNDING FROM X"

WHAT VALUE WOULD YOU GIVE TO THIS SITUATION?

*100000

WHAT ARE THE CHANCES OF THIS OUTCOME HAPPENING?

*0.4

WHAT IMMEDIATE COST IS EXPECTED?

*0

LET'S LOOK AT "PARTIAL FUNDING FROM X"

TRY TO PLACE A NUMERIC VALUE ON THIS SITUATION.

*70000

WHAT ARE THE CHANCES THAT THIS EVENT WILL OCCUR?

*0.4

WHAT WOULD BE THE IMMEDIATE COST, ASSUMING THIS SITUATION?

*0

WHAT ABOUT "X REJECTS PROPOSAL"?

HOW WOULD YOU EVALUATE THIS SITUATION?

*0

HOW MUCH EXPENSE IS ANTICIPATED?

*0

ARE YOU UNHAPPY WITH THE ACCURACY OF THESE PROBABILITY ESTIMATES?

*YES

WHAT EXPERIMENT COULD BE PERFORMED?

*EVALUATE THE PROPOSAL

WHEN WOULD IT BE PERFORMED?

*AFTER WAITING TWO MONTHS

PLEASE LIST THE POSSIBLE OBSERVATIONS FROM THE EXPERIMENT.

1.*FAVORABLE RESPONSE

2.*ADVERSE RESPONSE

3.*DONE

CAN JUST ONE OUTCOME HAPPEN AT A TIME?

*YES

SUPPOSE THAT "X REJECTS PROPOSAL" IS ABOUT TO HAPPEN...

WHAT IS THE PROBABILITY OF "FAVORABLE RESPONSE" HAPPENING?

*0.3

SUPPOSE THAT "PARTIAL FUNDING FROM X" IS ABOUT TO HAPPEN...

WHAT IS THE PROBABILITY OF "FAVORABLE RESPONSE" HAPPENING?

*0.9

SUPPOSE THAT "FULL FUNDING FROM X" IS ABOUT TO HAPPEN...

WHAT IS THE PROBABILITY OF "FAVORABLE RESPONSE" HAPPENING?

*1

ASSUME THAT "FULL FUNDING FROM X" HAS HAPPENED.

IS THERE A DECISION TO BE MADE AT THIS POINT?

*NO

ARE THERE SOME EVENTS THAT MAY HAPPEN?

*NO

PERHAPS WE SHOULD TALK ABOUT SOMETHING ELSE?

*YES

LET US SAY IT WAS "PARTIAL FUNDING FROM X" THAT ACTUALLY TOOK PLACE.

DO YOU HAVE A CHOICE OF ALTERNATIVES?

*NO

COULD UNCONTROLLABLE OUTCOMES OCCUR?

*NO

SHALL WE TERMINATE THE INTERVIEW?

*YES

CONCERNING SENDING THE PROPOSAL TO AGENCY Y OR WAITING

YOUR BEST DECISION IS TO WAIT FOR TWO MONTHS

EVALUATE THE PROPOSAL AND...

IF "FAVORABLE RESPONSE" IS OBSERVED, SEND TO X

IF "ADVERSE RESPONSE" IS OBSERVED, SEND TO Y

PROGRAM FINISHED

Event-Conditionals

P(favorable| win)=1
P(adverse| win)=0
P(favorable| partial)=.9
P(adverse| partial)=.1
P(favorable| lose)=.3
P(adverse| lose)=.7

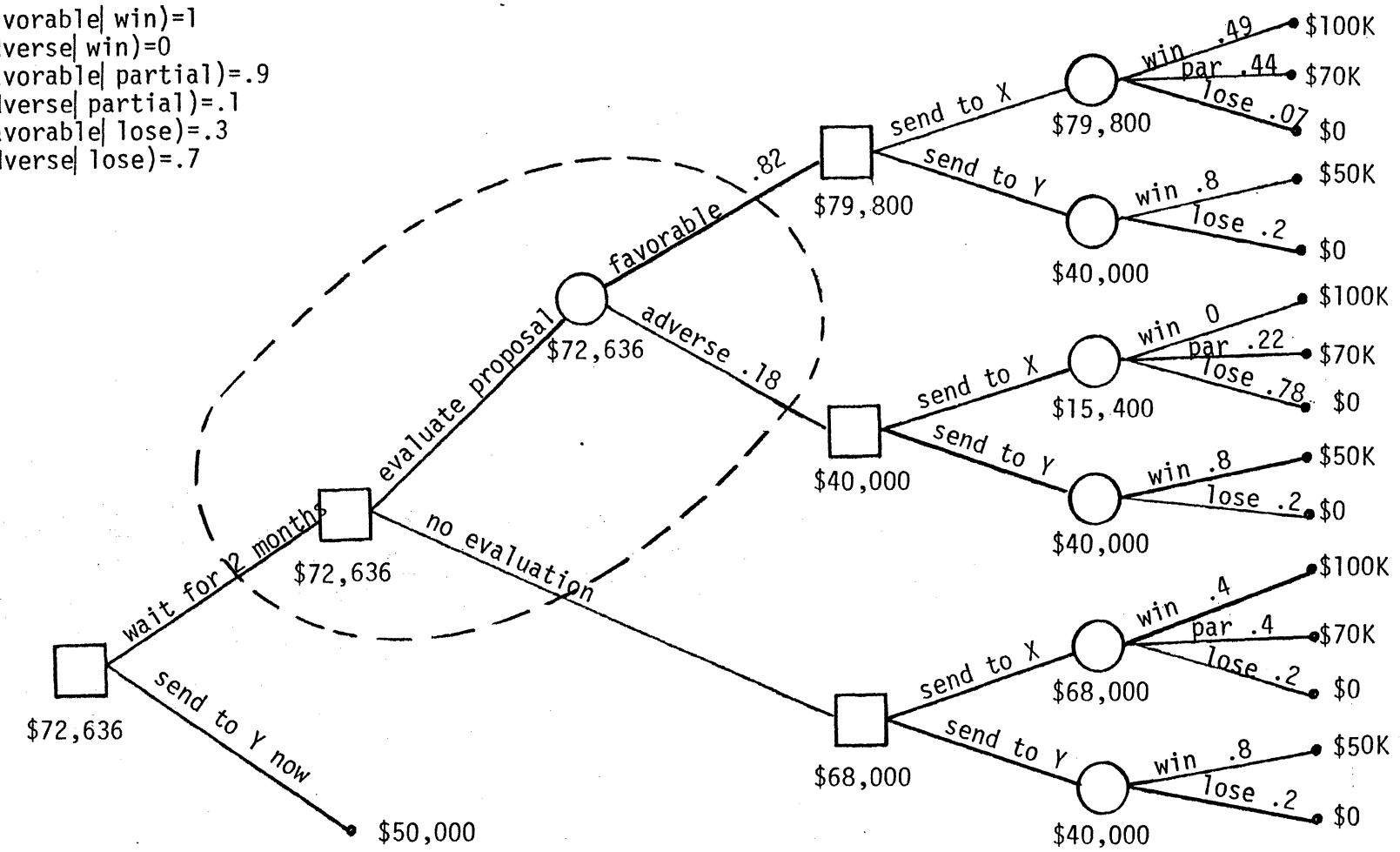


Figure 5.2. The Full Decision Tree

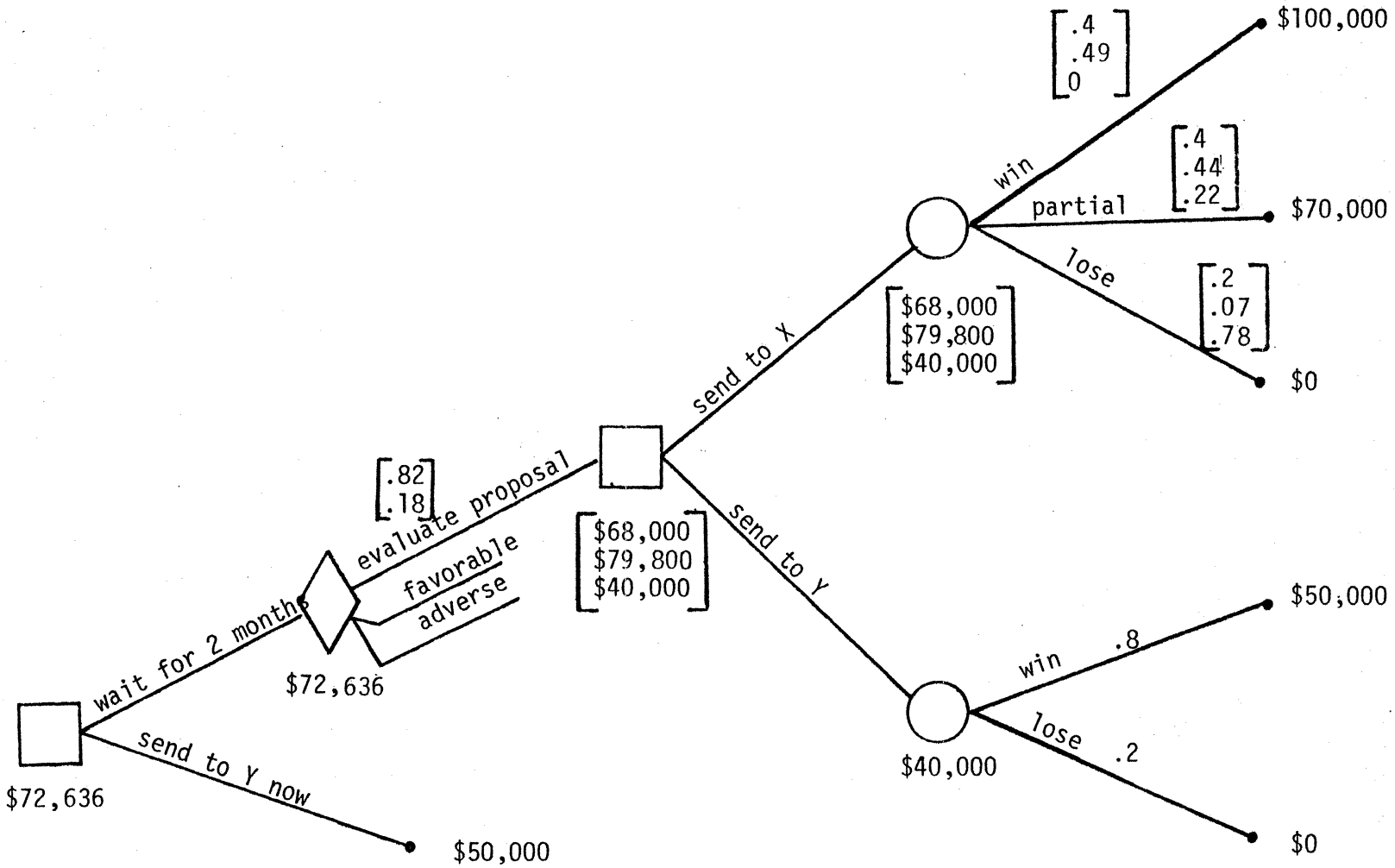


Figure 5.3. The Condensed Decision Structure

VI. PROGRAM DESCRIPTION

An overall diagram of the flow of information in the program is shown in Figure 6.1. The decision maker communicates with the system through a remote terminal display. Information is viewed on the screen and responses are typed at the console keyboard. Questions and answers are monitored by the "Query/Response Control" subsystem. It is responsible for activities such as message printing, information display, keyword analysis of input, etc., and the interface control between the decision maker and the main processing system. The "Tree Expansion Control" mechanism activates and maintains the primary tree expansion cycle of (1) rollback, (2) sensitivity analysis, (3) node selection, and (4) node expansion. Of these four major steps, the decision maker is only aware of the fourth, "node expansion", which guides him through the elicitation procedure. Each of the four major step-subsystems has access to the decision tree data but only the node expansion subsystem can alter it.

The program is implemented in the LISP programming language [14,15] on a Digital Equipment Corporation PDP-10 Computer [16] located in the Center for Computer-Based Behavioral Studies (CCBS) in Franz Hall on the UCLA Campus. The particular version of LISP that was used is called "Irvine LISP" [17] from the University of California at Irvine. It is a modified and enriched version of "Stanford LISP" [18]. A concerted effort was spent in an attempt to use as few sophisticated and version-dependent functions and special features

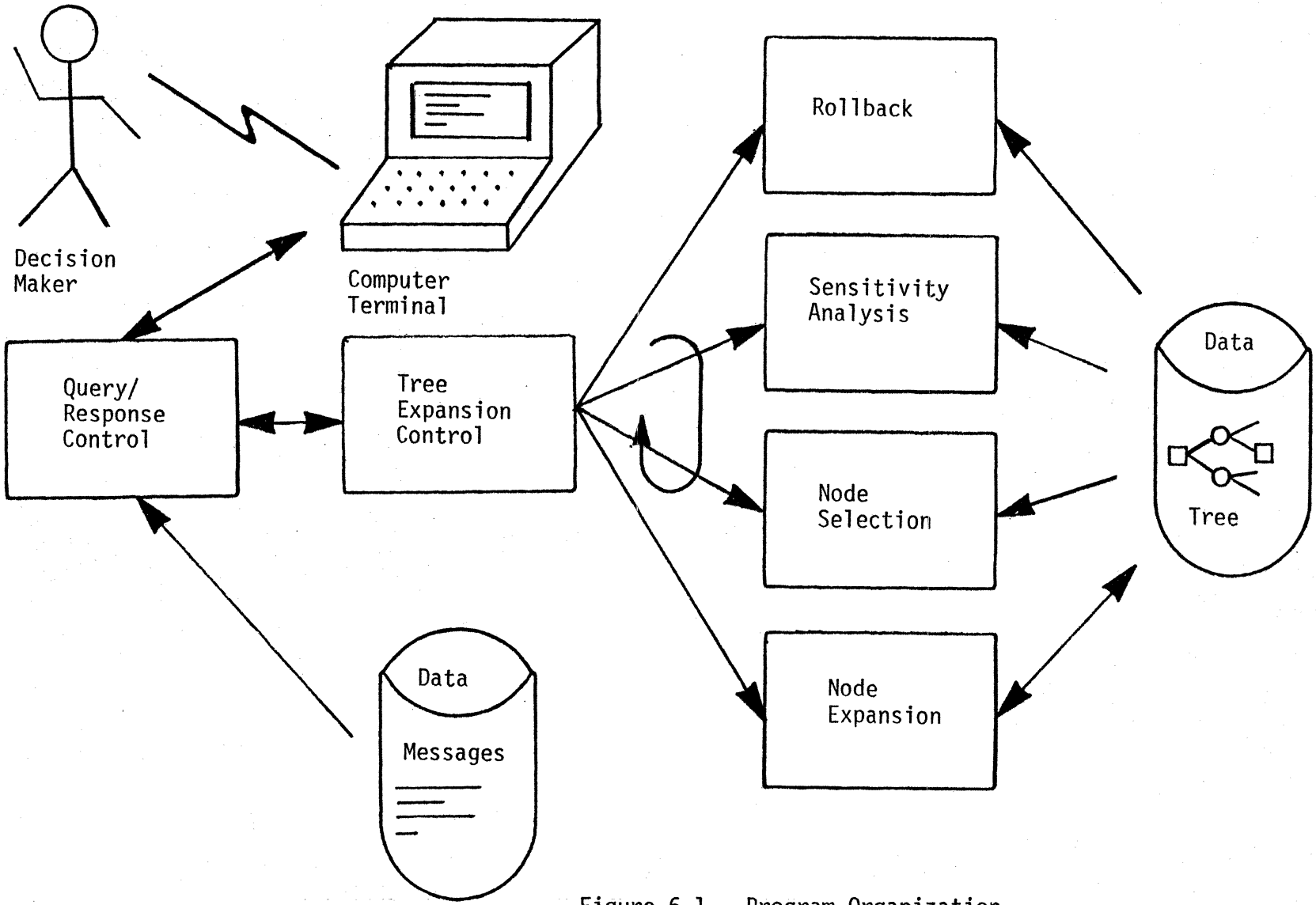
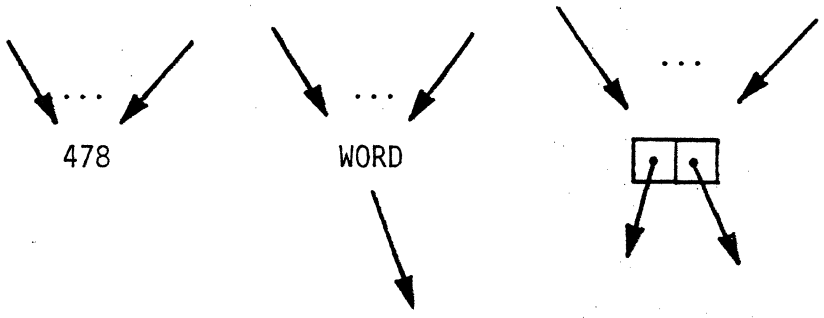


Figure 6.1. Program Organization

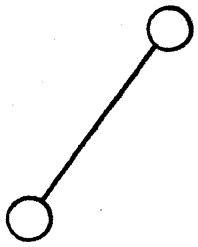
as possible. Thus, most of the program contains elements found in the intersection of all major LISP dialects. This allows the program to be transferred to another computer with a minimum of translation effort. The program itself contains a little over 100 defined functions and occupies almost 600 lines of printed LISP code. It does, however, execute rapidly as there is no detectable delay in response time. A complete program code listing can be found in the Appendix.

The basic data elements provided by the LISP programming language consist of atomic items such as numbers and words plus a single structural item called a binary cell. Each data element can be referenced by any number of "pointers"; words can, themselves, reference one other item; and binary cells can reference two items as shown below.

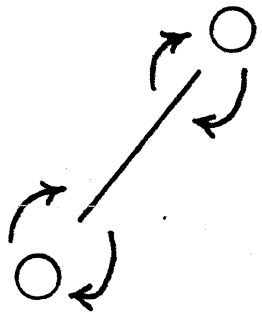


The data elements of the decision trees have been built from the above LISP primitive elements. The nodes and arcs of the tree are considered to be *individual entities* that are connected by pointers.

For example, two nodes connected by an arc are usually drawn as follows:



The above structure is, however, represented in data storage as follows:



Each arc is doubly connected to both its single predecessor and single successor node. Similarly, each node points to its predecessor arc and to all of its successor arcs as well as being pointed to by them.

The internal structure of each node permits the storage of necessary information for computational processing (see Figure 6.2). Each node has 9 data items associated with it. The NAME of the node is a list of words received from the decision maker. One binary cell is needed for each word in the node name. The BEST PATH FLAG is set to "true" when the node is found to be in the optimal solution plan

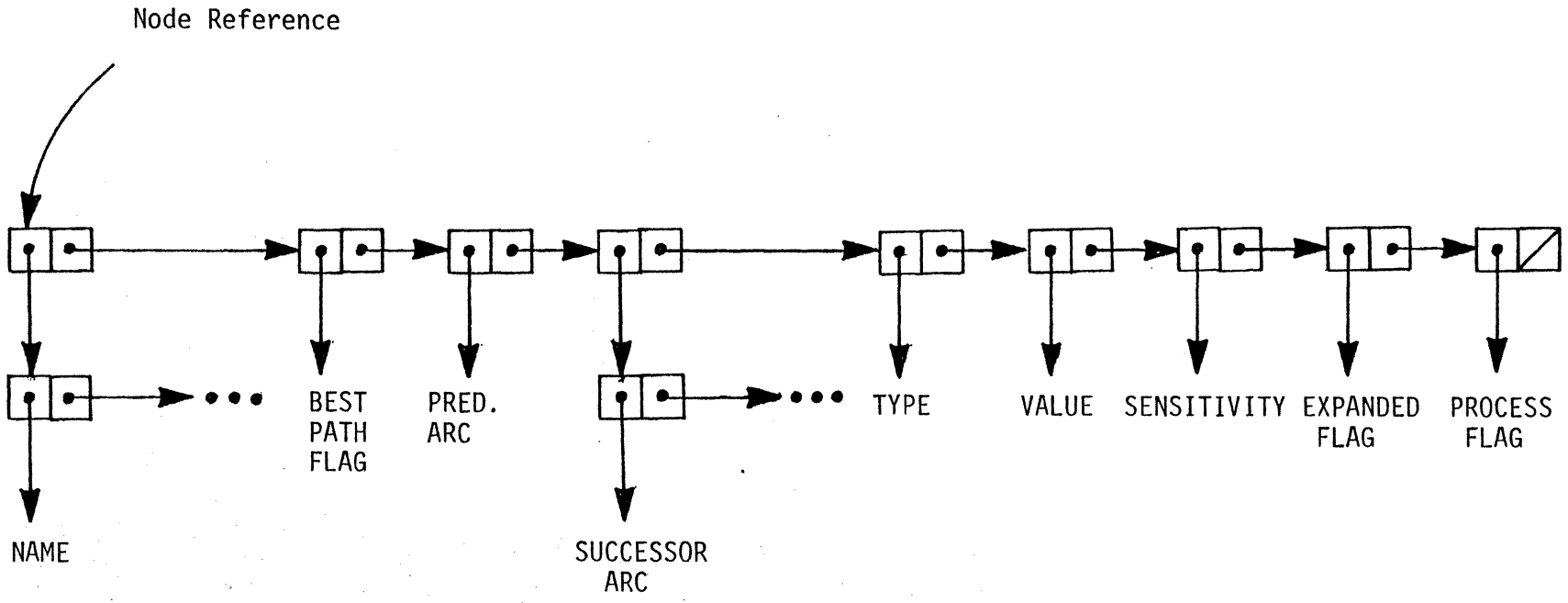


Figure 6.2. Internal Node Structure

after rollback calculations are completed. The PREDECESSOR ARC pointer connects directly to the arc structure that leads from the predecessor node. Similar provision is made for connection to multiple SUCCESSOR ARCS. The node TYPE is stored as either an integer from 0 to 3 or NIL indicating an unknown type. The following table lists the possible node types and their corresponding codes:

CODE	NODE TYPE
NIL	unknown
0	terminal
1	decision
2	event
3	experiment

A newly created node has type NIL until further information is acquired. The VALUE item is determined by the rollback procedure and may be a list of numbers if it is a collapsed node due to an experiment. Likewise, the SENSITIVITY is calculated for each value. The EXPANDED FLAG is set to "true" only when complete information about a node has been obtained. The PROCESS FLAG is an internal status indicator that is used by the rollback functions.

The internal structure of an arc parallels that of a node. Figure 6.3 shows this structure in detail. The data items are similar except for the SUCCESSOR NODE which connects to one node only. The PROBABILITY (which may be a vector) and the COST are obtained from the decision maker.

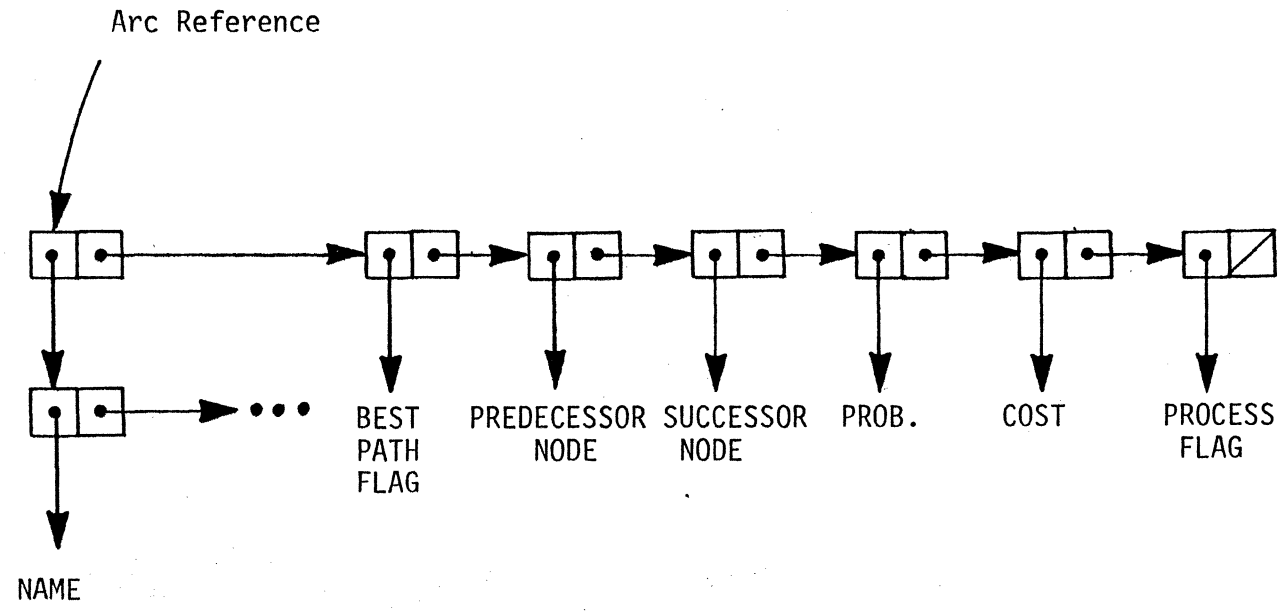


Figure 6.3. Internal Arc Structure

The node and arc structures are very flexible. All important information can be stored either at the node or on the arc (branch), wherever it is most natural. The tree can be searched either forward or backward as required. In addition, a "super-structure" of binary cells connects all nodes together as they are being created (see Figure 6.4). A similar one connects all arcs. These super-structures are used when node or arc processing is needed that is independent of tree structure.

The defined program functions are divided into four general categories: (1) data functions, (2) manipulation functions, (3) input/output functions, and (4) processing functions. The data functions are the lowest level functions and allow storage and retrieval from the data structures for node and arc information. The LISP language has perfectly good data functions of its own for this very same purpose. However, by defining new ones, two important advantages are gained: data independence and program readability. Data independence is gained when it is possible to change the data structure without changing the applications programs. When a second level of data functions is defined in terms of the primitive LISP functions, only their definitions need to be altered when a modification of data structure is required. Secondly, English words that reflect the meaning of the function can be used as names, thus making the program more readable. The following is a synopsis of the data functions and a short phrase describing the data item(s) that each retrieves.

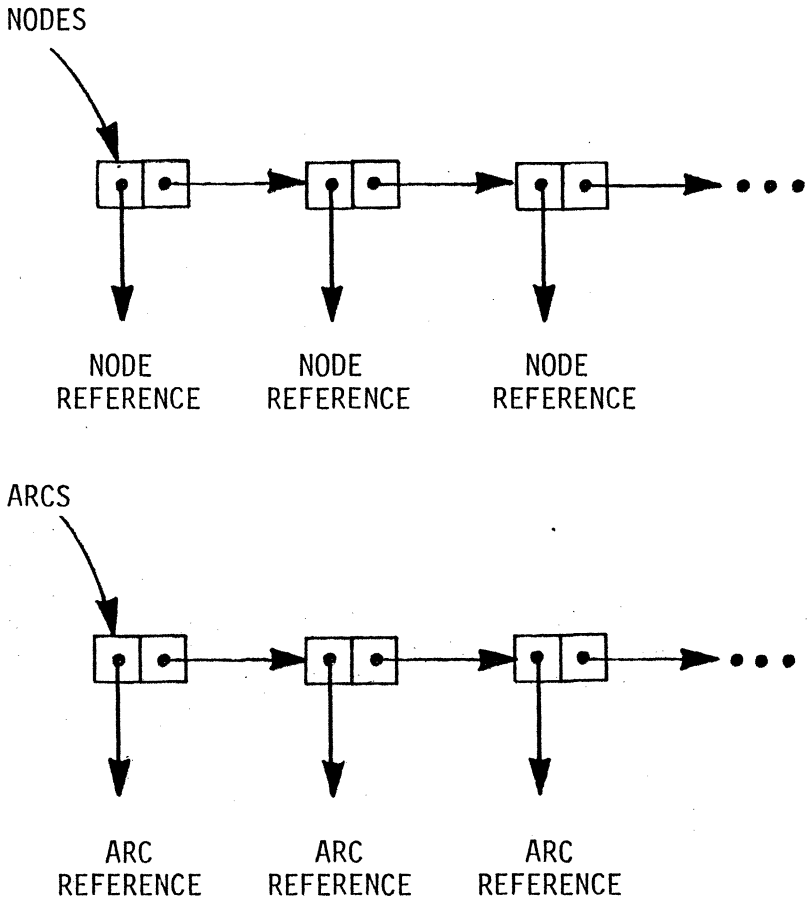


Figure 6.4. Node and Arc Super-Structures

NAME(X)	The words comprising the name of node X or arc X.
NAMES(L)	NAME applied to each node or arc in list L.
PUTNAME(X,V)	Stores a new name V at node X or arc X.
FINDANAME(N)	Searches node N or its predecessors for a name.
FINDNAMES(L)	FINDANAME applied to all nodes in list L.
NAME;(N)	Attaches a semi-colon to the end of the name of node N.
BP(X)	Contents of the BEST PATH FLAG cell.
PUTBP(X,V)	Places value V into BEST PATH FLAG cell of node or arc X.
PRED(X)	Predecessor arc of node X or predecessor node of arc X.
PUTPRED(X,V)	Stores V as a predecessor to node X or arc X.
PREDNODE(N)	PRED(PRED(N))
SUCC(X)	Successor arcs to node X or successor node to arc X.
PUTSUCC(X,V)	Stores V as a new successor to node or arc X.
SUCCNODES(N)	SUCC(SUCC(N))
ADDSUCC(N,V)	Adds a successor arc V to node N.
TYPE(N)	Numeric type of node N.
PUTTYPE(N,V)	Stores V as type of node N.
TYPE0(N)	"true" if type of node N is 0; NIL otherwise.
TYPE1(N)	"true" if type of node N is 1; NIL otherwise.
TYPE2(N)	"true" if type of node N is 2; NIL otherwise.
TYPE3(N)	"true" if type of node N is 3; NIL otherwise.

VALUE(N)	Provisional value of node N.
PUTVALUE(N,V)	Stores value V at node N.
VALUEO(N)	Applies VALUE to node N; if none stored, returns 0.
VALUES(L)	VALUE applied to all nodes in list L.
ADDVALUE(N,V)	Adds value V to those already at node N.
SEN(N)	Sensitivity of node N.
PUTSEN(N,V)	Places sensitivity value V at node N.
ADDSEN(N,V)	Adds sensitivity value V to those already at node N.
EXP(N)	EXPANDED FLAG cell contents.
EXPS(L)	EXP applied to all nodes in list L.
PUTEXP(N,V)	Stores value V in EXPANDED FLAG cell.
FLAG(X)	Contents of PROCESS FLAG cell.
PUTFLAG(X,V)	Stores V into PROCESS FLAG cell.
FLAGS(L)	FLAG applied to all nodes or arcs in list L.
PROB(A)	Probability of arc A.
PUTPROB(A,V)	Stores probability V at arc A.
ADDPROB(A,V)	Adds probability V to those already at arc A.
PROB1(A)	Applies PROB to arc A; if none stored, returns 1.
PROBS(L)	PROB applied to all arcs in list L.
PROB1S(L)	PROB1 applied to all arcs in list L.

COST(A)	Cost of arc A.
PUTCOST(A,V)	Stores cost V at arc A.
COSTO(A)	Applies COST to arc A; if none stored, returns 0.
COSTS(L)	COST applied to all arcs in list L.

The manipulation functions are defined to transform list structures without any particular application dependency. The LISP language has many available manipulation functions and relatively few new ones needed to be defined. Some of the functions operate on arbitrary lists and others are tailored for the specific decision tree structure. The definitions can be found in the Appendix.

FLAT(L) This function removes any hierarchy that may exist in the argument list L and returns a one-level list containing all of the atomic data items of the original list in their proper order. The scan algorithm searches depth-first from left-to-right.

SINGLE(L) This function removes all duplications of atomic items from a list at the top level only.

MEMBLISTV(L1,L2) This function returns "true" if any member of the list L1 is also a member of list L2.

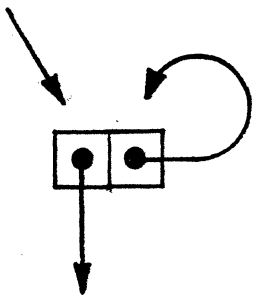
MEMBLIST&(L1,L2) Similarly, this function returns "true" only if *all* members of list L1 are also members of list L2. If any elements of list L1 are themselves lists, MEMBLISTV is called on those and vice

versa for MEMBLIST&. Thus, the two functions are mutually recursive.

BLEND(F V P C) For any function F, BLEND returns a list that is the computation $F(P_i(V_i-C_i))$ where P_i are probabilities in list P, V_i are values in list V, and C_i are costs in list C. This function is a useful way of computing rollback for different types of nodes using the same code.

ALLATOMS(L) This functions returns "true" only if all of the members of list L are atomic data items.

CONSTANT(S) A self-referent node pointing to S is formed when the function CONSTANT is used.



The reason that such a function is necessary stems from the pre-defined nature of the LISP function MAPCAR [17]. The function MAPCAR applies any specified function of one argument to the members of a given list (one at a time) and returns a list of results. If MAPCAR is given more than one list upon which to iterate, functions of more

than one argument may be supplied. In these cases, the elements of the multiple lists are taken in parallel until one of them runs out at which time MAPCAR halts. In many instances, it is desirable to give a *constant* argument to the specified function. Rather than creating a list of duplicates, a self-referent "list" fools MAPCAR into believing that an infinitely long list exists.

BAYES(A,B) Bayes' rule is used in this function to calculate observation-conditional probabilities.

FIRSTNUM(L) A simple function that returns the first number found in the argument list L.

STAT() The function STAT is a debugging tool that gives a status report on the values stored in the entire decision tree.

DUMP() The final summary report to the decision maker is initiated with this function.

MAXNODE(L) The node with the highest current provisional value in list L is returned as the value of MAXNODE.

TIPS(N) As might be expected, the function TIPS returns a list of tip nodes using node N as the root of a subtree.

The input/output functions control all of the interaction between the program and the decision maker. They consist of pre-defined messages (labelled M1 through M17), question functions (labelled with a prefix of "Q"), and various input/output processing functions. The message functions simply cycle through a list of equivalent messages printing a new one out each time it is needed.

M1	Alert (decision) message.
M2	Alert (event) message.
M3	Determine node type (decision).
M4	Determine node type (event).
M5	Determine node type (terminal).
M6	Elicit decision alternatives.
M7	Elicit event outcomes.
M8	Decision alternatives mutually exclusive?
M9	Event outcomes mutually exclusive?
M10	Successor iteration alert message.
M11	Elicit provisional value.
M12	Elicit probability.
M13	Elicit cost.
M14	Request for experiment.
M15	Request for experiment name.
M16	Request for experiment time.
M17	Elicit experiment conditional probabilities.

The question functions use the message functions and are labelled to refer to them. For example, question Q345 references message functions M3, M4, and M5.

QYN	Requests a "yes" or a "no" response.
Q345	Elicits node type.
Q67	Elicits decision alternatives or event outcomes.
Q89	Requests confirmation for mutually exclusive alternatives or outcomes.
Q11	Controls elicitation of provisional value.
Q12	Controls elicitation of probabilities.
Q13	Controls elicitation of cost.

The following input/output functions perform printing and reading chores, etc.

PR(LF L) The function PR prints atoms in a list with a trailing line-feed character if the variable LF is "true". Thus, it is possible to print messages from more than one place in the program on the same line.

OUT(V C L) The function OUT controls the cycle of messages and keeps track which message of a set is next to print.

GETINPUT() This is the only function that actually requests an input from the decision maker. It prints an asterisk as a signal.

The processing functions are the main control over tree construction and the elicitation procedure. They use all of the functions described thus far.

START() The top level function that is called to begin the program is START. It initializes all of the system variables and expands the root node. It then calls START* which begins the elicitation cycle. When the interview is completed, START evokes the summary function DUMP which gives the final results to the decision maker.

START*(N) The function START* controls the continuous cycle of rollback, sensitivity analysis, node selection, and node expansion. START* completes it job when the decision maker wishes to end the interview or when the node selection function indicates that all tree nodes have become terminal.

ROLLBACK() This function controls the rollback procedure and calls ROLLCALC for computation on individual nodes.

ROLLCALC(N) The rollback value of node N is calculated and stored in the internal structure representing the node itself.

SENSITIVITY() This function controls the sensitivity analysis for the entire tree. It calls SENCALC for the computation on a single node.

SENCALC(N) This function calculates the sensitivity value for the specific node argument N and stores it in the appropriate location for use by the node selection function.

NODE-SELECT(R) The node selection function chooses a new node for expansion out of the subtree with root node R by comparing the sensitivity values.

EXPAND(N) The function EXPAND is the heart of the program because it directly controls the logic for the elicitation procedure. It evokes other functions that print alert messages; elicit provisional values, probabilities, costs, etc.; request responses from the decision maker; and allocate storage for new nodes and arcs.

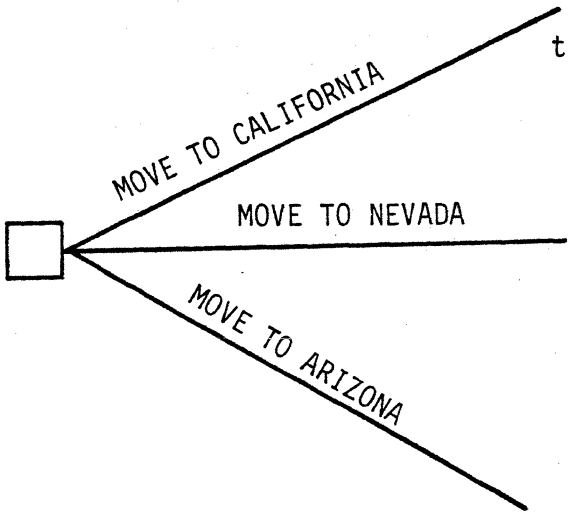
EXPERIMENT(N) This function is called by EXPAND when an experiment is desired by the decision maker. It controls the elicitation process for experiment nodes and inserts the node structure into the existing tree at the proper place.

FINDEXPNODE(F N) This function aids EXPERIMENT by locating the place of insertion of an experiment node. The method used is keyword matching of the words comprising the node names given by the decision maker with the name of the experiment.

VII. CONCLUSIONS

Although the program has not been in operation for a sufficient length of time to permit exhaustive tests, it was found that the elicitation system could provide an adequate and useful tool for decision aiding in those applications that naturally lent themselves to decision tree representations. The most desirable features were (1) the ease of operation that resulted once familiarity was gained with the mechanics of interaction; (2) the inherent "direction" of elicitation that leads to the exploration of the most important and appropriate areas for quickly resolving the major decision problem with few questions; and (3) the determination to maintain, as closely as possible, a one-to-one correspondence between the internal storage representation of events and the decision maker's perceptual representation.

This latter property is considered to be the most important and points out two observed deficiencies that will now be discussed in detail. The decision tree demands mutually exclusive decision alternatives and event outcomes. Thus, it is always necessary to confirm this fact by continually asking the decision maker to check his responses. This confirmation is necessary because of the ease with which the non-technical user may misinterpret the system's queries. It is very tempting to report *aspects* of the previously expanded node rather than new future situations. For example, suppose that the decision maker has just given the following alternatives to a decision node:



Assume that, through sensitivity analysis, it has been determined that tip node t is the most promising to expand. Further, the decision maker has indicated that t is an event node. Upon the request for the event outcomes of node t as follows:

EXACTLY WHAT EVENTS COULD HAPPEN?

the decision maker responds with this set of items:

1.* BUY A HOUSE

2.* GET AN APARTMENT

3.* RENT A ROOM

These responses constitute mutually exclusive event outcomes as

required by the decision structure and may be followed by the elicitation of probabilities. However, consider the following set:

1.* BUY A CAR

2.* GET AN APARTMENT

3.* FIND A JOB

These responses seem to follow just as naturally from the elicitation query and yet they are not mutually exclusive. Indeed, they are *aspects* or *attributes* of the previous decision alternative.

It would be desirable to incorporate these responses into the overall decision tree as a new type of node. There are, however, two distinct interpretations of the meaning of the outcomes and a separate query would be needed to distinguish them. The first interpretation is that they are a description of the event and, therefore, will occur with absolute certainty and serve primarily to identify the event. The reason that the decision maker might be tempted to provide a list of attributes in response to event queries is that he focuses on the node aspects as auxilliary tools for mental estimation of value (also called value dimensions). The second interpretation is that all possible combinations of the mentioned outcomes and their complements could be realized. In this case, the event node can be represented as a series of sequentially ordered events as shown in Figure 7.1, where P_i is the probability of event i occurring and $P_i' = 1-P_i$ stands for the probability of event i not occurring.

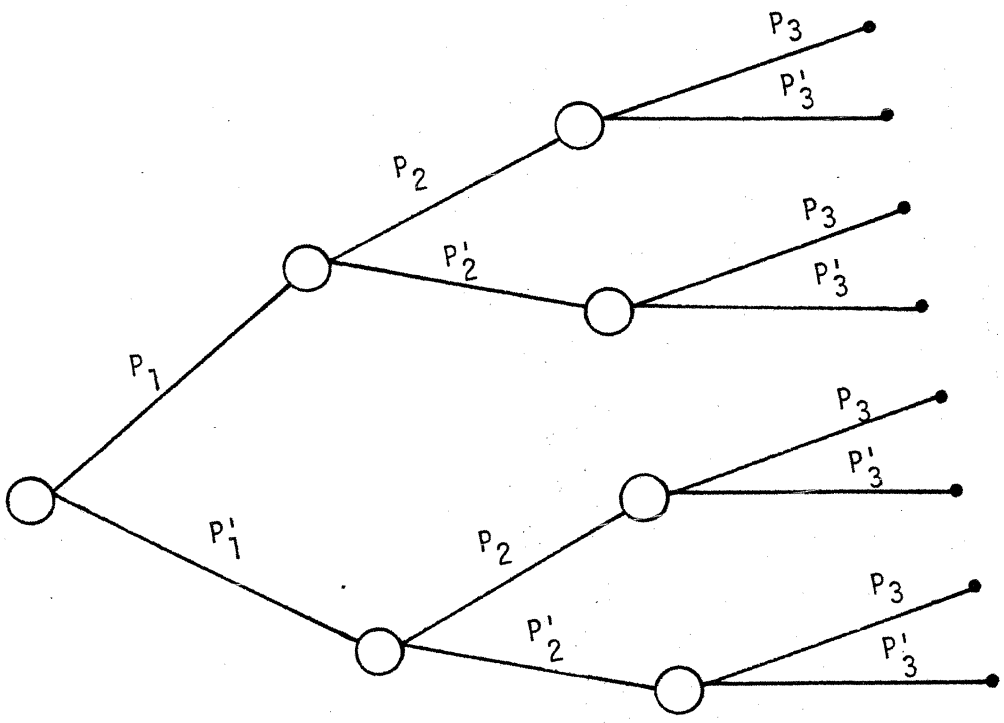


Figure 7.1. Interpretation of an Event Node

One immediate problem with this formulation is that the structure forces a duplication of event nodes as well as imposing a time ordering that was not implied by the responses. An alternate structure is shown in Figure 7.2 which requires only one node with more branches representing a set of mutually exclusive event combinations. A separate branch is needed for each path in the original structure. The probability assigned to each branch is the product of the probabilities along the path leading to the corresponding combination of events. This representation, however, introduces its own problems. Each tip node now stands for a *combination* of events and the property of one-to-one correspondence between events and nodes is lost. This property is one of the primary motivations for collapsing duplicate subtrees by using a single experiment node. In manual elicitation, the analyst can usually utilize his real-world knowledge to resolve these ambiguous situations.

A similar situation arises when non-mutually exclusive decision alternatives are encountered. Suppose, for example, that the decision maker responds with the following set of alternatives:

1.* INVEST IN STOCK A

2.* INVEST IN STOCK B

3.* INVEST IN STOCK C

It seems clear that unless there is a minimum purchase requirement, any combination of the stocks could be chosen for investment purposes

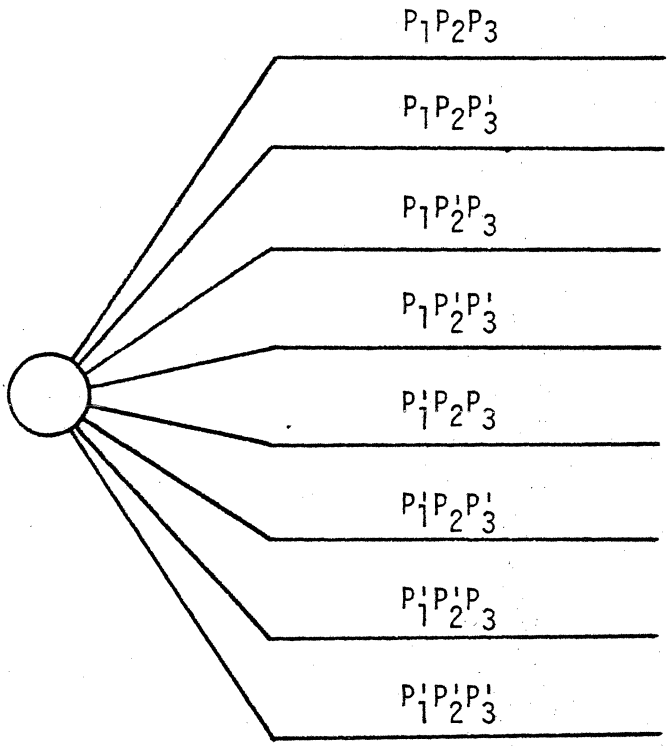


Figure 7.2. Order-independent, Non-duplicative Event Node

making the reported alternatives non-mutually exclusive. In fact, when considering the amount of capital invested, they form the boundary of a continuous action set.

The second deficiency is a subtle problem for any automated elicitation system and surfaces as a result of representing knowledge in tree form. In many real-world applications, the decision maker may not perceive a problem in the form of a time sequence of decision alternatives and event outcomes, but rather as a static network of influences surrounding *issues*.* Consider, for example, our perception of the environmental pollution problem. The issues of capital investment, energy needs, energy supply, unemployment, public health, etc. all seem to be tightly interwoven in a network of cause-and-effect relationships. The main difficulty in attacking such a problem seems to be that of unraveling the underlying causal network (on the basis of insufficient data) rather than that of planning with action/event scenarios. The major difference in the formal representation required for such problems and the one handled by decision trees is that the atomic entities admitted by the latter representation are restricted to be descriptions of "world states" or decision situations. The decision maker can express relations among these situation units but is unable to express relations between the constituents of these units. For example, he may assess the value of a certain situation consisting of a combination of attributes but he is unable (with the present system) to express the relative value of each individual attribute

*Ward Edwards elucidated this point in personal discussions with him.

or cross impacts among them. Likewise, when the decision maker is asked to assess the likelihood of a certain event taking place, he ought to consider the entire state of affairs prior to that occurrence. He cannot, for example, express the belief that pollution is a positive contributor to cancer independently of other situational factors such as unemployment or some energy embargo.

To facilitate an "issue-oriented" problem elicitation program, the internal machine representation of problem situations should follow a different structure. In the Artificial Intelligence literature, such structures are referred to as "problem reduction" representations [8]. Each node in a tree represents a sub-problem or a sub-goal rather than a state description. The resolution of each "issue", which the decision maker perceives as part of his problem, constitutes a reduction of the global problem into several components. These can be further reduced to their constituencies, and so on. The tree expansion procedure guides the decision maker in selecting the issue to explore next rather than the time-sequenced scenario that he should follow next.

It is believed that an amalgamation of the issue-oriented and the scenario-oriented approaches into a unified problem representation would yield greater matching between human perception and organization of information and would render computers a more effective decision aid to man.

REFERENCES

[1] Matheson, J.E. and R.A. Howard, "An Introduction to Decision Analysis", Readings in Decision Analysis, edited by R.A. Howard, Stanford Research Institute, Palo Alto, California, 1967.

[2] North, D.W., "A Tutorial Introduction to Decision Theory", IEEE Transactions on System Science and Cybernetics, Vol. SSC-4, No. 3, September 1968, pp. 200-210.

[3] Bellman, R.E. and L.A. Zadeh, "Decision-Making in a Fuzzy Environment", Management Science, Vol. 17, No. 4, December 1970.

[4] Geoffrion, A.M., J.S. Dyer, and A. Feinberg, "An Interactive Approach for Multi-Criterion Optimization with an Application to the Operation of Academic Departments", Management Science, Vol. 19, No. 4, December 1972.

[5] Dalkey, N., "Group Decision Analysis", School of Engineering and Applied Sciences, University of California at Los Angeles, UCLA-ENG-7571, August 1975.

[6] Raiffa, H., Decision Analysis, Addison Wesley, Reading, Massachusetts, 1970, pp. 7-34

[7] Pearl, J., "A Framework for Processing Value Judgments", School of Engineering and Applied Sciences, University of California at Los Angeles, UCLA-REP-7622, March 1976.

[8] Nilsson, N.J., Problem Solving Methods in Artificial Intelligence, McGraw Hill, New York, 1971, pp. 43-154.

[9] Sandewall, E., "Heuristic Search: Concepts and Methods", Artificial Intelligence and Heuristic Programming, edited by N.V. Findler, Americal Elsevier, New York, 1971, PP. 81-100.

[10] Edwards, W., C.D. Phillips, W.L. Hays, and B.G. Goodman, "Probabilistic Information Processing System", IEEE Transactions on System Science and Cybernetics, Vol. SSC-4, No. 3, September 1968.

[11] Pearl, J., "On the Complexity of Computing Probabilistic Assertions", Engineering Systems Department, University of California at Los Angeles, UCLA-ENG-7562, July 1975.

[12] McKendry, J.M., "Utility of Information as a Predictor of Decision Adequacy in Ambiguous Choice Situations", H.R.B. Singer, Inc., Report 567-R-3, July 1965.

[13] Galanter, E., "Utility Scales of Monetary and Non-monetary Events", Psychophysics Laboratory, Columbia University, New York, Report PLP-36, March 1975.

[14] Weissman, C., LISP 1.5 Primer, Dickenson Publishing Co., 1967.

[15] McCarthy, J., P.W. Abrahams, D.J. Edwards, T.P. Hart, and M.I. Levin, LISP 1.5 Programmer's Manual, 2nd ed., The MIT Press, Cambridge, Massachusetts, 1965.

[16] Digital Equipment Corporation, DEC System 10 User's Handbook, 2nd ed., Maynard Massachusetts, 1972.

[17] Bobrow, R.J., R.R. Burton, J.M. Jacobs, and D. Lewis, UCI LISP Manual, Department of Information and Computer Science, University of California at Irvine, Technical Report 21, 1973.

[18] Quam, L.H., Stanford LISP 1.6 Manual, Stanford Artificial Intelligence Laboratory, Palo Alto, California, Note 28.7.

APPENDIX

PROGRAM CODE LISTING

```
 1.   (SETQ BASE 12)(SETQ IBASE 12)
 2.   (DE NAME(X)(CAR X))
 3.   (DE NAMES(L)(MAPCAR(FUNCTION NAME)L))
 4.   (DE PUTNAME(X V)(CAR(RPLACA X V)))
 5.   (DE FINDANAME(N)(COND
 6.      ((NAME N))
 7.      ((NULL(PRED N))(QUOTE ROOT))
 8.      ((NAME(PRED N)))
 9.      (T(QUOTE NONE)) ))
10.   (DE FINDNAMES(L)(MAPCAR(FUNCTION FINDANAME)L))
11.   (DE NAME;(N)(LIST(NAME N);))
12.   (DE BP(N)(CADR N))
13.   (DE PUTBP(N V)(CAR(RPLACA(CDR N)V)))
14.   (DE PRED(X)(CADDR X))
15.   (DE PUTPRED(X V)(CAR(RPLACA(CDDR X)V)))
16.   (DE PREDNODE(N)(PRED(PRED N)))
17.   (DE SUCC(X)(CADDDR X))
18.   (DE PUTSUCC(X V)(CAR(RPLACA(CDDDR X)V)))
19.   (DE SUCCNODES(N)(MAPCAR(FUNCTION SUCC)(SUCC N)))
20.   (DE ADDSUCC(N V)(CAR(RPLACA(CDDDR N)(CONS V(SUCC N)))))
21.   (DE TYPE(N)(CAR(CDDDDR N)))
22.   (DE PUTTYPE(N V)(CAR(RPLACA(CDDDDR N)V)))
23.   (DE TYPE0(N)(EQ(TYPE N)0))
24.   (DE TYPE1(N)(EQ(TYPE N)1))
25.   (DE TYPE2(N)(EQ(TYPE N)2))
26.   (DE TYPE3(N)(EQ(TYPE N)3))
27.   (DE VALUE(N)(CADR(CDDDDR N)))
28.   (DE PUTVALUE(N V)(CAR(RPLACA(CDR(CDDDDR N))V)))
29.   (DE VALUE0(N)(COND((VALUE N))(T 0)))
30.   (DE VALUES(L)(MAPCAR(FUNCTION VALUE0)L))
```

```
31.   (DE ADDVALUE(N V)(PROGN
32.      (COND((ATOM(VALUE N))(PUTVALUE N(LIST(VALUE N)V)))
33.      (T(PUTVALUE N(REVERSE(CONS V(REVERSE(SEN N)))))) )))
34.   (DE SEN(N)(CADDR(CDDDDR N)))
35.   (DE PUTSEN(N V)(CAR(RPLACA(CDDR(CDDDDR N))V)))
36.   (DE ADDSEN(N V)(PROGN
37.      (COND((ATOM(SEN N))(PUTSEN N(LIST(SEN N)V)))
38.      (T(PUTSEN N(REVERSE(CONS V(RESERSE(SEN N)))))) )))
39.   (DE EXP(N)(CADDDR(CDDDDR N)))
40.   (DE EXPS(L)(MAPCAR(FUNCTION EXP)L))
41.   (DE PUTEXP(N V)(CAR(RPLACA(CDDDR(CDDDDR N))V)))
42.   (DE FLAG(X)(CAR(LAST X)))
43.   (DE PUTFLAG(X V)(CAR(RPLACA(LAST X)V)))
44.   (DE FLAGS(L)(MAPCAR(FUNCTION FLAG)L))
45.   (DE OBN(A)(CADR A))
46.   (DE PUTOBN(A V)(CAR(RPLACA(CDR A)V)))
47.   (DE PROB(A)(CAR(CDDDDR A)))
48.   (DE PUTPROB(A V)(CAR(RPLACA(CDDDDR A)V)))
49.   (DE ADDPROB(A V)(PROGN
50.      (COND((ATOM(PROB A))(PUTPROB A(LIST(PROB A)V)))
51.      (T(PUTPROB A(REVERSE(CONS V(REVERSE(PROB A)))))) )))
52.   (DE PROB1(A)(COND((PROB A))(T 1)))
53.   (DE PROBS(L)(MAPCAR(FUNCTION PROB)L))
54.   (DE PROB1S(L)(MAPCAR(FUNCTION PROB1)L))
55.   (DE COST(A)(CADR(CDDDDR A)))
56.   (DE PUTCOST(A V)(CAR(RPLACA(CDR(CDDDDR A))V)))
57.   (DE COSTO(A)(COND((COST A))(T 0)))
58.   (DE COSTS(L)(MAPCAR(FUNCTION COSTO)L))
59.   (DE FLAT(L)(PROG(R)
60.      (FLAT* L)
```

```
61.       (RETURN(REVERSE R)) ))
62.    (DE FLAT*(L)(COND
63.       ((NULL L)NIL)
64.       ((ATOM L)(SETQ R(CONS L R)))
65.       (T(PROGN
66.          (FLAT*(CAR L))
67.          (FLAT*(CDR L)) ))))
68.    (DE SINGLE(L)(COND
69.       ((NULL L)NIL)
70.       ((MEMBER(CAR L)(CDR L))(SINGLE(CDR L)))
71.       (T(CONS(CAR L)(SINGLE(CDR L))))))
72.    (DE FIRSTNUM(L)(COND
73.       ((FIRSTNUM* L))
74.       (T L) ))
75.    (DE FIRSTNUM*(L)(COND
76.       ((NULL L)NIL)
77.       ((NUMBERP(CAR L))(CAR L))
78.       (T(FIRSTNUM*(CDR L)))))
79.    (DE SETQS()(PROGN
80.       (SETQ COSTFLAG(SETQ ENDFLAG NIL))
81.       (SETQ Q11FLAG(SETQ Q12FLAG(SETQ Q13FLAG(SETQ NFLAG(SETQ AFLAG()))))))
82.       (SETQ M19C(SETQ M18C(SETQ M17C(SETQ M16C(SETQ M15C(SETQ M14C
83.       (SETQ M13C(SETQ M12C(SETQ M11C(SETQ M10C(SETQ M9C
84.       (SETQ M8C(SETQ M7C(SETQ M6C(SETQ M5C(SETQ M4C
85.       (SETQ M3C(SETQ M2C(SETQ M1C 0)))))))))))))))))))
86.       (SETQ /.(QUOTE /.))
87.       (SETQ /,(QUOTE /,))
88.       (SETQ ?(QUOTE ?))
89.       (SETQ *(QUOTE *))
90.       (SETQ '(QUOTE '))
```

```
91.      (SETQ :(QUOTE :))
92.      (SETQ %(QUOTE %))
93.      (SETQ #(QUOTE #))
94.      (SETQ SP(QUOTE / ))
95.      (SETQ ;(QUOTE ;))
96.      (SETQ /"(QUOTE /"))
97.      (SETQ \(QUOTE  \))
98.      (SETQ =(QUOTE =))
99.      (SETQ LF T)
100.     (SETQ NODES(LIST(SETQ ROOT(LIST
101.        NIL NIL NIL NIL 1 NIL NIL NIL NIL))))
102.     (SETQ ARCS(SETQ PROBLEM NIL))
103.     (QUOTE SETQS) ))
104.  (DE STAT()(PROG(AFLAG NFLAG)
105.     (PRINT(QUOTE ARCS))
106.     (MAPCAR(FUNCTION ASTAT)ARCS)
107.     (PRINT(QUOTE NODES))
108.     (MAPCAR(FUNCTION NSTAT)NODES)
109.     (RETURN(QUOTE DONE)) ))
110.  (DE NSTAT(N)(PROG()
111.     (COND(NFLAG(RETURN))
112.          ((EQUAL(GETINPUT)(QUOTE(SKIP)))(RETURN(SETQ NFLAG T))))
113.     (PR LF(LIST(QUOTE NAME=)(NAME N)))
114.     (PR LF(LIST(QUOTE BPF=)(BP N)))
115.     (PR LF(LIST(QUOTE PREDECESSOR=)(COND((PRED N)(NAME(PRED N)))
116.        (T NIL)) ))
117.     (PR LF(LIST(QUOTE SUCCESSORS=)(COND((SUCC N)(MAPCAR
118.        (FUNCTION NAME;)(SUCC N)))(T NIL))))
119.     (PR LF(LIST(QUOTE TYPE=)(SELECTQ(TYPE N)
120.        (O(QUOTE TERMINAL))
```

```
121.          (1(QUOTE DECISION))
122.          (2(QUOTE EVENT))
123.          (3(QUOTE EXPERIMENT))
124.          (QUOTE UNKNOWN))))
125.       (PR LF(LIST(QUOTE VALUE=)(VALUE N)))
126.       (PR LF(LIST(QUOTE SENSITIVITY=)(SEN N)))
127.       (PR LF(LIST(QUOTE EXPANDED=)(EXP N)))
128.       (PR LF(LIST(QUOTE FLAG=)(FLAG N))) ))
129.    (DE ASTAT(A)(PROG()
130.       (COND(AFLAG(RETURN))
131.             ((EQUAL(GETINPUT)(QUOTE(SKIP)))(RETURN(SETQ AFLAG T))))
132.       (PR LF(LIST(QUOTE NAME=)(NAME A)))
133.       (PR LF(LIST(QUOTE OBSERVATIONS=)(OBN A)))
134.       (PR LF(LIST(QUOTE PREDECESSOR=)(NAME(PRED A))))
135.       (PR LF(LIST(QUOTE SUCCESSOR=)(NAME(SUCC A))))
136.       (PR LF(LIST (QUOTE PROBABILITY=)(PROB A)))
137.       (PR LF(LIST(QUOTE COST=)(COST A)))
138.       (PR LF(LIST(QUOTE FLAG=)(FLAG A))) ))
139.    (DE DUMP()(PROGN
140.       (PR LF(LIST(QUOTE CONCERNING)PROBLEM))
141.       (PR LF(LIST(QUOTE(YOUR BEST DECISION IS TO))
142.          (FINDANAME(MAXNODE(SUCCNODES ROOT)1)) ))
143.       (MAPCAR(FUNCTION PLAN)(SUCCNODES ROOT)) ))
144.    (DE MAXNODE(L C)(PROG(N)
145.       (COND((NULL L)(RETURN)))
146.       (SETQ N(CAR L))
147.       L1 (COND((LESSP(PICKV N C)(PICKV(CAR L)C))(SETQ N(CAR L))))
148.       (COND((SETQ L(CDR L))(GO L1)))
149.       (RETURN N) ))
150.    (DE PICKV(N C)(COND
```

```
151.      ((ATOM(VALUEO N))(VALUEO N))
152.      (T(CAR(NTH(VALUE N)C))) ))
153.   (DE MINIL(X Y)(COND(Y(MIN X Y))(T X)))
154.   (DE TIPS(N)(PROG(TPS)
155.      (TIPS* N)
156.      (RETURN TPS) ))
157.   (DE TIPS*(N)(COND
158.      ((NULL(SUCC N))(SETQ TPS(CONS N TPS)))
159.      (T(MAPCAR(FUNCTION TIPS*)(SUCCNODES N)))))
160.   (DE MEMBLISTV(L1 L2)(COND
161.      ((NOT(AND L1 L2))NIL)
162.      ((AND(ATOM(CAR L1))
163.           (MEMB(CAR L1)L2))T)
164.      ((AND(NOT(ATOM(CAR L1)))
165.           (MEMBLIST&(CAR L1)L2))T)
166.      (T(MEMBLISTV(CDR L1)L2))))
167.   (DE MEMBLIST&(L1 L2)(COND
168.      ((NULL L2)NIL)
169.      ((NULL L1)T)
170.      ((AND(ATOM(CAR L1))
171.           (MEMB(CAR L1)L2))(MEMBLIST&(CDR L1)L2))
172.      ((AND(NOT(ATOM(CAR L1)))
173.           (MEMBLISTV(CAR L1)L2))(MEMBLIST&(CDR L1)L2))
174.      (T NIL)))
175.   (DE RESHUFFLE(L)(MAPCAR(FUNCTION EVAL)
176.      (MAPCAR(FUNCTION CONS)(CONSTANT(QUOTE PLUS))(REFORM
177.      (MAPCAR(FUNCTION(LAMBDA(L)(MAPCAR(FUNCTION(LAMBDA(X Y)(TIMES X Y)))
178.         (CONSTANT(CAR L))(CDR L)) ))L) ))))
179.   (DE BLEND(F V P C)(EVAL(CONS F(MAPCAR(FUNCTION(LAMBDA(V P C)
180.      (TIMES(DIFFERENCE V C)P)))V P C))))
```

```
(DE ALLATOMS(L)(EVAL(CONS(QUOTE AND)(MAPCAR(FUNCTION ATOM)L))))
(DE PLAN(N)(PROG(S SS)
   (SETQ S(CAR(SUCCNODES N)))
   (SETQ SS(CAR(SUCC N)))
   (COND((NULL N)(RETURN))
        ((AND(TYPE1 N)(ATOM(VALUE N)))(PR LF(LIST(QUOTE IF) /" #
           (NAME(PRED N)) # /" (COND((TYPE1(PRED N))(QUOTE OCCURS/,))
              (T(QUOTE(IS CHOSEN/,))))(NAME (PRED
           (MAXNODE(SUCCNODES N)1))) )))
        ((AND(TYPE3 N)(TYPE1 S))
           (COND((GREATERP(CAR(VALUE S))(BLEND(FUNCTION PLUS)
                   (CDR(VALUE S))(PROB SS)(CONSTANT 0)))
                   (PR LF(LIST(QUOTE(DO NOT))(NAME SS)
                   (QUOTE BUT)(NAME(PRED(MAXNODE(SUCCNODES S)1))))))
                 (T(PROG(C)
                      (PR LF(LIST(NAME SS)(QUOTE AND/./.)))
                      (SETQ C 1)
                      L1 (PR LF(LIST SP SP SP SP SP(QUOTE IF) /" #
                          (CAR(NTH(OBN SS)C)) # /" (QUOTE(IS OBSERVED/,))
                          (NAME(PRED(MAXNODE(SUCCNODES S)(PLUS C 1)))) ))
                      (COND((GREATERP(SETQ C(PLUS C 1))(LENGTH(OBN SS)))
                              (RETURN)))
                      (GO L1) )))))
   (MAPCAR(FUNCTION PLAN)(SUCCNODES N)) ))
(DE TRANSPOSE(L)(PROG(X)
   (SETQ X L)
   L1 (COND((NOT(ATOM(CAR X)))(GO L2))
            ((SETQ X(CDR X))(GO L1))
            (T(RETURN L)))
   L2 (SETQ X(CAR X))
```

```
211.      (SETQ L(REMOVE X L))
212.      (RETURN(MAPCAR(FUNCTION CONS)X(CONSTANT L))) ))
213.    (DE PR(LF L)(PROG()
214.      (COND((NULL L)(RETURN)))
215.      (SETQ L(FLAT L))
216.      L1 (COND((NOT(MEMB NIL L))(GO L2)))
217.      (SETQ L(REMOVE NIL L))
218.      (GO L1)
219.      L2 (COND((NULL L)(RETURN(COND(LF(TERPRI))(T))))
220.               ((EQ(CAR L)#)(PROGN(SETQ L(CDR L))(GO L3)))
221.               ((EQ(CAR L)\)(PROGN(SETQ L(CDR L))(TERPRI))))
222.      (PRINC SP)
223.      L3 (PRINC(CAR L))
224.      (SETQ L(CDR L))
225.      (GO L2) ))
226.    (DE OUT(V C L)(PROG()
227.      (SET V(COND((EQ(EVAL V)C)1)(T(PLUS(EVAL V)1))))
228.      (PR LF(CAR(NTH L(EVAL V)))) ))
229.    (DE M1(X)(OUT(QUOTE M1C)5(LIST
230.     (LIST(QUOTE(SUPPOSE THAT YOU HAD CHOSEN TO))X # /.)
231.     (LIST(QUOTE(ASSUMING THAT))/" # X # /"(QUOTE(WAS PICKED/,)) )
232.     (LIST(QUOTE(SUPPOSE THAT))/" # X # /"(QUOTE(IS YOUR CHOICE/.)) )
233.     (LIST(QUOTE(WHAT IF YOU CHOOSE TO)) X #  ?)
234.     (LIST(QUOTE(LET US SAY THAT YOU TOOK))/" # X # /" # /,)
235.      )))
236.    (DE M2(X)(OUT(QUOTE M2C)4(LIST
237.     (LIST(QUOTE(SUPPOSE THAT))/" # X # /"(QUOTE HAPPENED/.) )
238.     (LIST(QUOTE(WHAT IF))/" # X # /"(QUOTE OCCURS?))
239.     (LIST(QUOTE(ASSUME THAT))/" # X # /"(QUOTE(HAS HAPPENED/,)) )
240.     (LIST(QUOTE(LETS SAY IT WAS))/" # X # /"(QUOTE(THAT ACTUALLY
```

```
241.        TOOK PLACE/.)))
242.      )))
243.    (DE M3()(OUT(QUOTE M3C)4(LIST
244.     (QUOTE(IS THERE A DECISION TO BE MADE AT THIS POINT?))
245.     (QUOTE(DO YOU HAVE A CHOICE OF ALTERNATIVES?))
246.     (QUOTE(ARE THERE SOME OPTIONS AVAILABLE TO YOU?))
247.     (QUOTE(WOULD THERE BE OPPORTUNITIES OPEN TO YOU NOW?))
248.       )))
249.    (DE M4()(OUT(QUOTE M4C)4(LIST
250.     (QUOTE(CAN YOU THINK OF THINGS THAT MAY HAPPEN  AS A
251.            RESULT?))
252.     (QUOTE(ARE EVENTS ABOUT TO HAPPEN OVER WHICH YOU HAVE NO CONTROL?))
253.     (QUOTE(ARE THERE SOME EVENTS THAT MAY HAPPEN?))
254.     (QUOTE(COULD UNCONTROLLABLE OUTCOMES OCCUR?))
255.       )))
256.    (DE M5()(OUT(QUOTE M5C)5(LIST
257.     (QUOTE(DO YOU WISH TO STOP EXPLORING FURTHER IN THIS DIRECTION?))
258.     (QUOTE(HAS THERE BEEN ENOUGH DETAIL EXPRESSED SO FAR?))
259.     (QUOTE(SHOULD WE EXPLORE SOME OTHER POSSIBILITIES IN ANOTHER AREA?))
260.     (QUOTE(PERHAPS WE SHOULD TALK ABOUT SOMETHING ELSE?))
261.     (QUOTE(I ASSUME THAT WE CAN LEAVE THIS SITUATION?))
262.       )))
263.    (DE M6()(OUT(QUOTE M6C)6(LIST
264.    (QUOTE(PLEASE LIST THE ALTERNATIVES THAT YOU HAVE/, ONE AT A TIME/.))
265.     (QUOTE(STATE THE CHOICES THAT YOU HAVE/.))
266.     (QUOTE(LIST THE OPTIONS THAT ARE AVAILABLE/.))
267.     (QUOTE(PLEASE LIST THE DECISIONS THAT YOU COULD MAKE/.))
268.     (QUOTE(TELL ME WHAT IS AVAILABLE/.))
269.     (QUOTE(ENTER THE CHOICES THAT YOU COULD TAKE/.))
270.     (QUOTE(WHAT OPPORTUNITIES ARE THERE?))
```

```
271.      )))
272.    (DE M7()(OUT(QUOTE M7C)5(LIST
273.     (QUOTE(PLEASE LIST THE OUTCOMES/.))
274.     (QUOTE(EXACTLY WHAT EVENTS COULD OCCUR?))
275.     (QUOTE(STATE THOSE EVENTS THAT MAY HAPPEN/.))
276.     (QUOTE(LIST THE POSSIBLE OCCURANCES THAT YOU FORESEE/.))
277.     (QUOTE(WHAT DO YOU SUPPOSE COULD HAPPEN?))
278.      )))
279.    (DE M8()(OUT(QUOTE M8C)5(LIST
280.     (QUOTE(ARE THE ALTERNATIVES MUTUALLY EXCLUSIVE?))
281.     (QUOTE(DOES THE CHOICE OF ONE ALTERNATIVE EXCLUDE THE OTHERS?))
282.     (QUOTE(IS ONLY ONE CHOICE POSSIBLE?))
283.     (QUOTE(I ASSUME THAT ONLY ONE OF THE DECISIONS CAN
284.            BE MADE/, CORRECT?))
285.     (QUOTE(IS IT TRUE THAT CHOOSING ONE OF THESE ALTERNATIVES
286.            EXCLUDES THE OTHERS FROM BEING CHOSEN?))
287.      )))
288.    (DE M9()(OUT(QUOTE M9C)4(LIST
289.     (QUOTE(ARE THESE EVENTS MUTUALLY EXCLUSIVE?))
290.     (QUOTE(DOES THE OCCURANCE OF ONE EVENT EXCLUDE THE OTHERS
291.            \ FROM HAPPENING?))
292.     (QUOTE(CAN JUST ONE OUTCOME HAPPEN AT A TIME?))
293.     (QUOTE(I ASSUME THAT ONLY ONE CAN OCCUR/, CORRECT?))
294.      )))
295.    (DE M10(X)(OUT(QUOTE M10C)7(LIST
296.     (LIST(QUOTE(LETS CONSIDER))/" # X # /"(QUOTE(FOR A MOMENT/.)))
297.     (LIST(QUOTE(NOW CONSIDER))/" # X # /" # /.)
298.     (LIST(QUOTE(LETS LOOK AT))/" # X # /" /.)
299.     (LIST(QUOTE(WHAT ABOUT))/" # X # /" # ?)
300.     (LIST(QUOTE(LETS TALK ABOUT))/" # X # /"(QUOTE(FOR AWHILE/.)))
```

```
301.     (LIST(QUOTE(NOW CONSIDER))/" # X # /" # /.)
302.     (LIST(QUOTE(NOW CONSIDER))/" # X # /" # /.)
303.       )))
304.    (DE M11()(OUT(QUOTE M11C)5(LIST
305.     (QUOTE(WHAT VALUE WOULD YOU GIVE TO THIS SITUATION?))
306.     (QUOTE(TRY TO PLACE A NUMERIC VALUE ON THIS SITUATION/.))
307.     (QUOTE(HOW WOULD EVALUATE THIS SITUATION?))
308.     (QUOTE(ESTIMATE THE VALUE IF YOU WERE IN THIS POSITION/.))
309.     (QUOTE(WHAT IS THE SITUATION WORTH TO YOU?))
310.       )))
311.    (DE M12()(OUT(QUOTE M12C)5(LIST
312.     (QUOTE(TRY TO ESTIMATE THE PROBABILITY THAT THIS EVENT WILL HAPPEN/.))
313.     (QUOTE(WHAT ARE THE CHANCES OF THIS OUTCOME HAPPENING?))
314.     (QUOTE(WHAT ARE THE CHANCES THAT THIS EVENT WILL OCCUR?))
315.     (QUOTE(HOW MUCH OF A CHANCE DOES IT HAVE?))
316.     (QUOTE(WHAT PROBABILITY WOULD YOU PLACE ON IT?))
317.       )))
318.    (DE M13()(OUT(QUOTE M13C)5(LIST
319.     (QUOTE(WHAT IMMEDIATE COST IS EXPECTED?))
320.     (QUOTE(WHAT WOULD BE THE IMMEDIATE COST/, ASSUMING THIS SITUATION?))
321.     (QUOTE(HOW MUCH EXPENSE IS ANTICIPATED?))
322.     (QUOTE(IF THERE IS AN ASSOCIATED COST/, WHAT IS IT?))
323.     (QUOTE(STATE THE IMMEDIATE COST/, IF THERE IS ONE/.))
324.       )))
325.    (DE QYN(L)(COND
326.       ((MEMBLISTV(QUOTE(Y YES OK RIGHT YEA))L)T)
327.       ((MEMBLISTV(QUOTE(N NO NONE NOTHING DONE 0))L)NIL)
328.       (T L)))
329.    (DE Q89(N)(PROG(ANS)
330.       L1(COND((TYPE1 N)(M8))(T(M9)))
```

```
331.      (COND((NOT(ATOM(SETQ ANS(QYN(GETINPUT)))))(GO L1))
332.            ((NOT ANS)(PROGN
333.              (PR LF(QUOTE(TRY TO LIST MUTUALLY EXCLUSIVE ITEMS/.)))
334.              (RETURN T) )))
335.      (RETURN) ))
336.    (DE Q345(S)(PROG(ANS)
337.      (COND((GREATERP S 2)(SETQ S(DIFFERENCE S 2))))
338.      L1 (COND((EQ S 1)(M4))
339.               ((EQ S 2)(M3))
340.               (T(ERROR-Q345-1)))
341.      (COND((NOT(ATOM(SETQ ANS(QYN(GETINPUT)))))(GO L1))
342.            (ANS(PROGN(SETQ ENDFLAG NIL)
343.                 (RETURN(SELECTQ S(1 2)(2 1)(ERROR-Q345-2))))))
344.      L2 (COND((EQ S 1)(M3))
345.               ((EQ S 2)(M4))
346.               (T(ERROR-Q345-3)))
347.      (COND((NOT(ATOM(SETQ ANS(QYN(GETINPUT)))))(GO L2))
348.            (ANS(PROGN(SETQ ENDFLAG NIL)(RETURN S))))
349.      L3 (M5)
350.      (COND((NOT(ATOM(SETQ ANS(QYN(GETINPUT)))))(GO L3))
351.            ((NOT ANS)(GO L4)))
352.      (COND((NOT ENDFLAG)(SETQ ENDFLAG T))
353.            (T(PROGN(PR LF(QUOTE(SHALL WE TERMINATE THE INTERVIEW?)))
354.                    (SETQ ENDFLAG(COND((QYN(GETINPUT))(QUOTE STOP))
355.                                      (T NIL))))))
356.      (RETURN 0)
357.      L4 (PR LF(QUOTE(TRY TO PREDICT WHETHER YOU HAVE CONTROL OVER FUTURE
358.                    \ EVENTS OR WHETHER THEY WILL OCCUR BY THEMSELVES/.
359.                    \ ENTER 0 1 OR 2:
360.                    \ 0 IF YOU WISH TO STOP/.
```

```
361.                \ 1 IF YOU HAVE A DECISION TO MAKE/.
362.                \ 2 IF UNCONTROLLABLE EVENTS WILL OCCUR/.)))
363.      (SETQ ANS(CAR(GETINPUT)))
364.      (COND((OR(EQ ANS 0)(EQ ANS 1)(EQ ANS 2))(RETURN ANS)))
365.      (PR LF(QUOTE(LETS MOVE ON TO ANOTHER AREA/.)))
366.      (RETURN 0) ))
367.   (DE Q67()(PROG(C S)
368.      (SETQ C 1)
369.      L1 (PR NIL C)
370.      (COND((NULL(QYN(CAR(SETQ S(CONS(GETINPUT)S)))))
371.         (RETURN(REVERSE(CDR S))) ))
372.      (SETQ C(PLUS 1 C))
373.      (GO L1) ))
374.   (DE Q11()(PROG(ANS)
375.      (COND((NULL Q11FLAG)(GO L3)))
376.      L1 (M11)
377.      L2 (SETQ ANS(FIRSTNUM(GETINPUT)))
378.      (COND((ATOM ANS)(RETURN ANS))(T(GO L1)))
379.      L3 (PR LF(QUOTE(ASSUMING THAT THE CURRENT SITUATION WERE TRUE/, \
380.         TRY TO RATE YOUR ESTIMATION OF HOW GOOD IT IS/, \
381.         INDEPENDENT OF OTHER POSSIBILITIES/. YOU MAY USE \
382.         MONEY AS A SCALE OR ONE OF YOUR OWN CHOOSING/. HOWEVER/, \
383.         YOU MUST BE CONSISTENT AND USE THE SAME SCALE THROUGHOUT \
384.         THE ENTIRE INTERVIEW/. \ DO YOU WISH TO USE MONEY AS A SCALE?)))
385.      (SETQ COSTFLAG(QYN(GETINPUT)))
386.      (PR LF(QUOTE(ENTER YOUR ESTIMATE/.)))
387.      (SETQ Q11FLAG T)
388.      (GO L2) ))
389.   (DE Q12()(PROG(ANS)
390.      (M12)
```

```
391.      (RETURN(FIRSTNUM(GETINPUT))) ))
392.    (DE Q12*(L)(DIFFERENCE 1.0(EVAL(CONS(QUOTE PLUS)(REMOVE NIL L)))))
393.    (DE Q13()(PROG(ANS)
394.       (COND((NULL Q13FLAG)(GO L3)))
395.       L1 (M13)
396.       L2 (SETQ ANS(FIRSTNUM(GETINPUT)))
397.       (COND((ATOM ANS)(RETURN ANS))(T(GO L1)))
398.       L3 (PR LF(QUOTE(ARE THERE ANY HIDDEN COSTS NOT TAKEN INTO \
399.          ACCOUNT IN THE PREVIOUS VALUE ESTIMATION?)))
400.       (SETQ Q13FLAG T)
401.       (COND((NOT(QYN(GETINPUT)))(RETURN 0)))
402.       (GO L2) ))
403.    (DE GETINPUT()(PROG(L)
404.       L1 (SETQ L(LINEREAD))
405.       (COND((NOT(MEMBER(QUOTE LISP)L))(GO L2)))
406.       (PRINT(QUOTE LISP:))
407.       (PRINT(EVAL(LINEREAD)))
408.       (TERPRI)
409.       (GO L1)
410.       L2 (RETURN L) ))
411.    (DE M14()(OUT(QUOTE M14C)3(LIST
412.     (QUOTE(CAN YOU IMPROVE THESE PROBABILITY ASSIGNMENTS BY \
413.           PERFORMING AN EXPERIMENT?))
414.     (QUOTE(ARE YOU UNHAPPY WITH THE ACCURACY OF THESE
415.      PROBABILITY ESTIMATES?))
416.     (QUOTE(IS IT POSSIBLE TO DO ANYTHING THAT WILL IMPROVE \
417.      THESE PROBABILITY VALUES?))
418.       )))
419.    (DE M15()(OUT(QUOTE M15C)1(LIST
420.     (QUOTE(WHAT EXPERIMENT COULD BE PERFORMED?))
```

```
421.       )))
422.    (DE M16()(OUT(QUOTE M16C)1(LIST
423.     (QUOTE(WHEN WOULD IT BE PERFORMED?))
424.       )))
425.    (DE M17()(OUT(QUOTE M17C)1(LIST
426.     (QUOTE(PLEASE LIST THE POSSIBLE OBSERVATIONS FROM THE EXPERIMENT/.))
427.       )))
428.    (DE M18(X)(OUT(QUOTE M18C)1(LIST
429.     (LIST(QUOTE(SUPPOSE THAT))/" # X # /"(QUOTE(IS ABOUT TO
430.       HAPPEN/././.)))
431.       )))
432.    (DE M19(X)(OUT(QUOTE M19C)1(LIST
433.     (LIST(QUOTE(WHAT IS THE PROBABILITY OF))/" # X # /"
434.       (QUOTE HAPPENING?))
435.       )))
436.    (DE EXPAND(N)(PROG(S CH NEWARC NEWNODE)
437.       (COND((NULL(PRED N))(GO L0)))
438.       ((SELECTQ(SETQ S(TYPE(PREDNODE N)))
439.          (1(FUNCTION M1))
440.          (2(FUNCTION M2))
441.          (QUOTE ERROR-EXPAND))
442.             (FINDANAME N))
443.       (PUTTYPE N(Q345 S))
444.       L0 (COND((TYPE0 N)(GO L4))
445.            ((TYPE1 N)(M6))
446.            (T(M7)) )
447.       (SETQ S(Q67))
448.       L1 (COND((Q89 N)(GO L0)))
449.       (SETQ M10C 1)
450.       L2 (SETQ ARCS(CONS(SETQ NEWARC(LIST
```

```
451.          (CAR S)NIL N NIL NIL NIL NIL ))ARCS))
452.        (ADDSUCC N NEWARC)
453.        (M10(CAR S))
454.        (SETQ NODES(CONS(SETQ NEWNODE(LIST
455.          NIL NIL NEWARC NIL NIL(Q11)NIL NIL NIL))NODES))
456.        (PUTSUCC NEWARC NEWNODE)
457.        L3 (COND((TYPE2 N)(PUTPROB NEWARC(COND((CDR S)(Q12))
458.                                  (T(Q12*(PROBS(SUCC N))))) )))
459.        (COND(COSTFLAG(PUTCOST NEWARC(Q13))))
460.        (COND((SETQ S(CDR S))(GO L2)))
461.        (COND((TYPE2 N)(EXPERIMENT N)))
462.        L4 (RETURN(PUTEXP N T)) ))
463.     (DE START()(PROGN
464.        (SETQS)
465.        (PR NIL(QUOTE(WHAT/'S YOUR PROBLEM?)))
466.        (SETQ PROBLEM(GETINPUT))
467.        (EXPAND ROOT)
468.        (MAPCAR(FUNCTION EXPAND)(SUCCNODES ROOT))
469.        (START* ROOT)
470.        (DUMP)
471.        (QUOTE "PROGRAM FINISHED") ))
472.     (DE START*(N)(PROG()
473.        (COND((NULL(EXP N))(EXPAND N)))
474.        L1 (ROLLBACK)
475.        (SENSITIVITY)
476.        (EXPAND(COND((NODE-SELECT N))(T(RETURN))))
477.        (COND((EQ(QUOTE STOP)ENDFLAG)(RETURN)))
478.        (GO L1) ))
479.     (DE ROLLBACK()(PROG()
480.        (MAPCAR(FUNCTION(LAMBDA(N)(PUTFLAG N
```

```
481.          (OR(NULL(TYPE N))(TYPEO N)))))NODES)
482.       L1 (COND((EVAL(CONS(QUOTE OR)(MAPCAR(FUNCTION ROLLCALC)
483.             NODES)))(GO L1))) ))
484.    (DE NODE-SELECT(SS)(PROG(N S)
485.       (SETQ S (TIPS SS))
486.       L1 (COND((NULL S)(COND(N(RETURN N))(T(GO L2))))
487.                ((AND(NULL(SEN(CAR S)))(NOT(TYPEO(CAR S))))
488.                     (RETURN(CAR S)) )
489.                ((OR(TYPEO(CAR S))(NULL(SEN(CAR S)))(ZEROP(SEN(CAR S))))NIL)
490.                ((OR(NULL N)(LESSP(SEN(CAR S))(SEN N)))
491.                     (SETQ N(CAR S)) ))
492.       (SETQ S(CDR S))
493.       (GO L1)
494.       L2 (SETQ S(TIPS SS))
495.       L3 (COND((AND(ZEROP(SEN(CAR S)))(NOT(TYPEO(CAR S))))(RETURN(CAR S)))
496.                ((SETQ S(CDR S))(GO L3)))
497.       (RETURN) ))
498.    (DE SENSITIVITY()(PROG(M V)
499.       (MAPCAR(FUNCTION(LAMBDA(X)(PUTSEN X NIL)))NODES)
500.       (SETQ M(EVAL(CONS(QUOTE MAX)(SETQ V(VALUES(SUCCNODES ROOT))))))
501.       (MAPCAR(FUNCTION SENCALC)
502.          (SUCCNODES ROOT)
503.          (CONSTANT 1.0)
504.          (MAPCAR(FUNCTION(LAMBDA(X)(DIFFERENCE M X)))V)) ))
505.    (DE SENCALC(N P DR)(PROG(SN)
506.       (PUTSEN N(SETQ SN(QUOTIENT DR P)))
507.       (COND((NULL(SUCC N))(RETURN)))
508.       (MAPCAR(FUNCTION SENCALC)
509.          (SUCCNODES N)
510.          (COND((TYPE1 N)(CONSTANT 1.0))
```

```
511.              ((TYPE2 N)(MAPCAR(FUNCTION(LAMBDA(X Y)(TIMES X Y)))
512.                      (PROB1S(SUCC N))
513.                    (CONSTANT P))))
514.          (COND((TYPE2 N)(CONSTANT DR))
515.                ((TYPE1 N)(MAPCAR(FUNCTION(LAMBDA(X)
516.                   (DIFFERENCE(PLUS SN(VALUE N))X)))
517.                   (VALUES(SUCCNODES N)) ))) )))
518.    (DE CONSTANT(X)(PROG(XX)(RETURN(RPLACD(SETQ XX(LIST X))XX))))
519.    (DE ROLLCALC(N)(PROG(V P C F)
520.       (COND((EVAL(CONS(QUOTE OR)(CONS(FLAG N)
521.          (MAPCAR(FUNCTION NOT)(FLAGS(SUCCNODES N))))))(RETURN)))
522.       (SETQ V(VALUES(SUCCNODES N)))
523.       (SETQ P(PROB1S(SUCC N)))
524.       (SETQ C(COSTS(SUCC N))) .
525.       (SETQ F(COND((TYPE1 N)(QUOTE MAX))(T(QUOTE PLUS))))
526.       (PUTVALUE N(COND
527.          ((TYPE3 N)(MAX(CAAR V)(BLEND(QUOTE PLUS)(CDAR V)
528.             (CAR P)(CONSTANT(CAR C)))))
529.          ((AND(ATOM(CAR P))(ALLATOMS V))(BLEND F V P C))
530.          (T(MAPCAR(FUNCTION BLEND)(CONSTANT F)
531.            (COND((TYPE1 N)(TRANSPOSE V))(T(CONSTANT V)))
532.            (COND((TYPE1 N)(CONSTANT P))(T(REFORM P)))
533.              (CONSTANT C) )) ))
534.       (RETURN(PUTFLAG N T)) ))
535.    (DE REFORM(L)(MAPCAR(FUNCTION(LAMBDA(C)
536.       (MAPCAR(FUNCTION CAR)
537.          (MAPCAR(FUNCTION NTH)L(CONSTANT C)) )))
538.       (REVERSE(MAPLIST(FUNCTION LENGTH)(CAR L))) ))
539.    (DE EXPERIMENT(N)(PROG(NN W EXPARC EXPNODE C EE E)
540.       LO (M14)
```

```
541.      (COND((NOT(ATOM(SETQ NN(QYN(GETINPUT))))))(GO L0))
542.           ((NOT NN)(RETURN)))
543.      (M15)
544.      (SETQ NN(GETINPUT))
545.      (M16)
546.      (SETQ W(FINDEXPNODE(GETINPUT)N))
547.      (SETQ EXPARC(LIST NN NIL NIL(SUCC W)NIL NIL NIL))
548.      (SETQ EXPNODE(LIST NIL NIL W(LIST EXPARC)3 NIL NIL NIL NIL))
549.      (PUTPRED(SUCC EXPARC)EXPARC)
550.      (PUTPRED EXPARC EXPNODE)
551.      (PUTSUCC W EXPNODE)
552.      (SETQ NODES(CONS EXPNODE NODES))
553.      (SETQ ARCS(CONS EXPARC ARCS))
554.      (M17)
555.      (SETQ C(SUCC N))
556.      L3 (SETQ EE(PUTOBN EXPARC(Q67)))
557.      (COND((Q89 N)(GO L3)))
558.      L1 (SETQ E EE)
559.      (M18(NAME(CAR C)))
560.      L2 (M19(CAR E))
561.      (ADDPROB(CAR C)(FIRSTNUM(GETINPUT)))
562.      (COND((SETQ E(CDR E))(COND((CDR E)(GO L2))
563.         (T(ADDPROB(CAR C)(DIFFERENCE 1.0(EVAL(CONS(QUOTE PLUS)
564.            (CDR(PROB(CAR C))))))))))
565.      (COND((SETQ C (CDR C))(GO L1)))
566.      (PUTPROB EXPARC(RESHUFFLE(PROBS(SUCC N))))
567.      (MAPCAR(FUNCTION PUTPROB)(SUCC N)
568.         (MAPCAR(FUNCTION CONS)(MAPCAR(FUNCTION CAR)
569.         (PROBS(SUCC N)))(MAPCAR(FUNCTION BAYES)
570.         (PROBS(SUCC N))(CONSTANT(PROB EXPARC)) )))
```

```
571.      (RETURN) ))
572.    (DE FINDEXPNODE(FIND N)(PROG(ARC SCORE HARC HSCORE WORDS)
573.      (SETQ HARC(SETQ ARC(PRED N)))
574.      (SETQ HSCORE 0)
575.      L1 (SETQ WORDS(NAME ARC))
576.      (SETQ SCORE 0)
577.      L2 (COND((MEMBER(CAR WORDS)FIND)(SETQ SCORE(PLUS 1 SCORE))))
578.      (COND((SETQ WORDS(CDR WORDS))(GO L2))
579.            ((LESSP SCORE HSCORE)(GO L3)))
580.      (SETQ HSCORE SCORE)
581.      (SETQ HARC ARC)
582.      L3 (COND((SETQ ARC(PRED(PRED ARC)))(GO L1)))
583.      (RETURN(COND((ZEROP HSCORE)NIL)(T HARC))) ))
584.    (DE BAYES(A B)(MAPCAR(FUNCTION(LAMBDA(X Y Z)
585.      (QUOTIENT(TIMES X Y)Z)))(CDR A)(CONSTANT(CAR A))B))
```