3

A DECISION TREE LANGUAGE

Fred M. Tonge and Roger Woodsmall

TECHNICAL REPORT No. 3, MARCH 1970

1. INTRODUCTION

This paper describes a language for creating, modifying, and manipulating decision trees. The language is intended as a tool for students and decision-makers in exploring the use of decision trees as a structure for stating and analyzing decision problems. It is intended to be implemented on a computer system with typewriter or teletype-like access. The language reflects the viewpoint of Raiffa, Howard, Decision Analysis, Addison-Wesley, 1968, and is perhaps best used in conjunction with that book.

The paper presents first an example of the use of the language in structuring a decision problem, then an informal description of the language, then a further example of its use, and finally some proposed extension to the language. Two technical appendices contain a precise and formal description of the language and some comments on a particular computer system implementation.

2. EXAMPLE

Decision trees are a means of displaying a decision problem so as to make clear the structure of the problem and the computations necessary to arrive at a solution.

As an example, consider the following problem (adapted from Raiffa). There is a collection of 1000 unmarked urns, 800 of which (type 1 urns) contain 4 red balls and 6 black balls and 200 of which (type 2 urns) contain 9 red balls and 1 black ball. A single urn is selected at random from the collection, and you as the decision-maker are offered the following bet. You are to guess whether the selected urn is type 1 or type 2. If you guess type 1 and the urn is type 1, you win $40, but if it is type 2 you lose $20. On the other hand, if you guess type 2 and are right, you win $100; in that case, if you are wrong you lose $5. You can, of course, refuse to bet. And, as an added inducement, for a modest fee of $8, you can draw one ball from the selected urn before guessing its type. What should you do?

One representation of this problem as a decision tree is given in figure 1. Squares indicate points at which the decision-maker has a choice; circles indicate points at which the decision is determined by chance. Dollar values at the ends of paths indicate the payoff if that particular sequence of choices and chance events should occur, and the probabilities associated with paths from chance events are the probability of that chance event given the previous events in the path to that event.

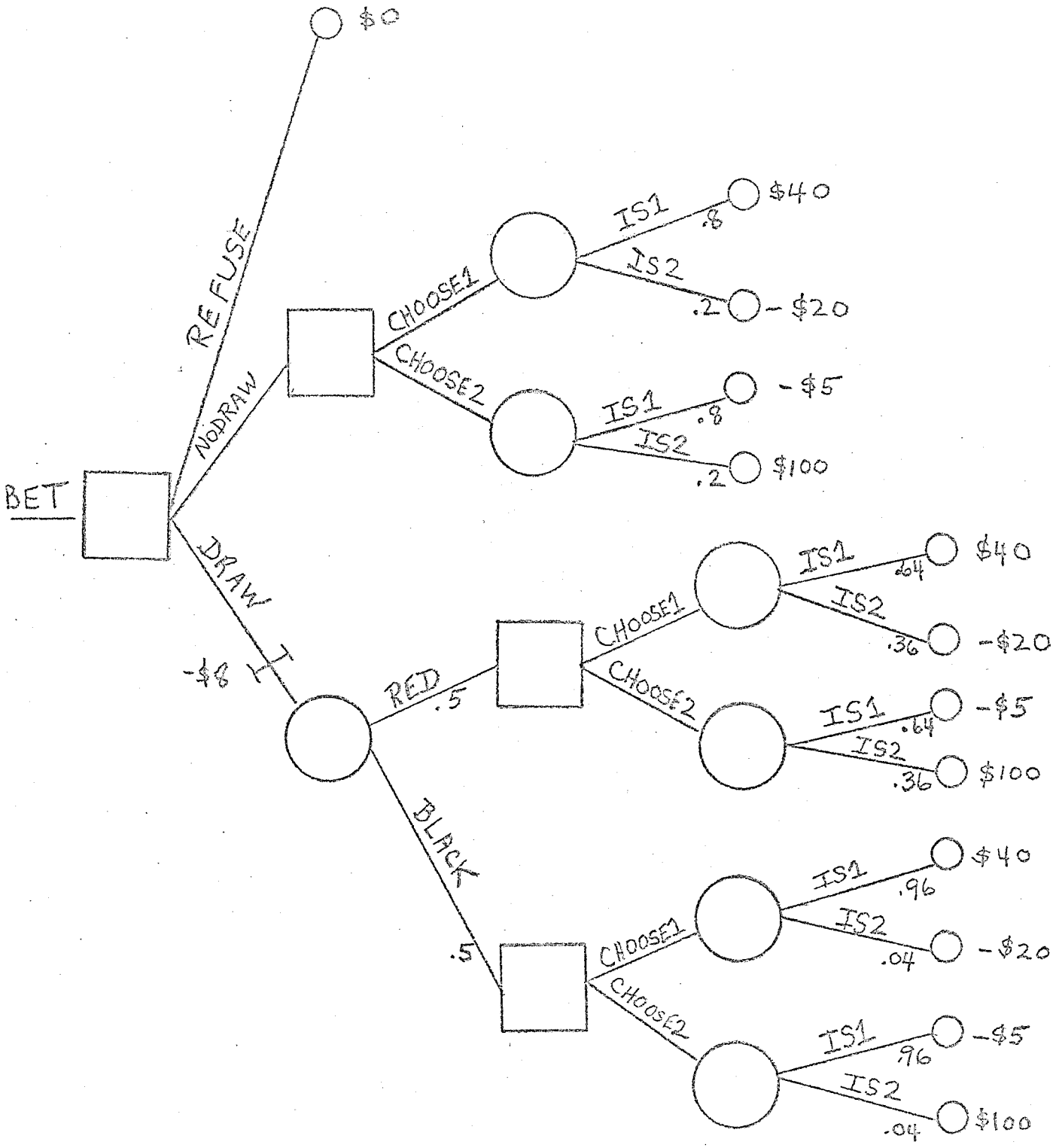

BET
REFUSE
$0
NODRAW
CHOOSE1
IS1
.8
$40
IS2
.2
-$20
CHOOSE2
IS1
.8
-$5
IS2
.2
$100
DRAW
-$6
RED
.5
CHOOSE1
IS1
.64
$40
IS2
.36
-$20
CHOOSE2
IS1
.64
-$5
IS2
.36
$100
BLACK
.5
CHOOSE1
IS1
.96
$40
IS2
.04
-$20
CHOOSE2
IS1
.96
-$5
IS2
.04
$100

Figure 1

The analysis of such a decision tree is carried out by successively (1) averaging the payoffs associated with the branches from a chance point to find an expected payoff, and (2) choosing from among the branches at a choice point that with the highest expected payoff (taking into account, of course, any extra charge for making a particular choice). This process of analysis is called "averaging out and folding back".

Figure 2 shows the result of such analysis for our example. On the average, the opportunity to engage in this wager should result in a profit of $28.00 for the decision-maker. He should not choose to pay $8.00 and draw a ball from the selected urn, but rather should always guess that the urn is type 1. The right to draw a ball is no bargain at $8.00, but it would be at any charge less than $7.20.

(We shall omit here any discussion of whether it is reasonable, or even permissible, to use averages and expected values in making a choice that is not repeated. Raiffa discusses this point at some length.)

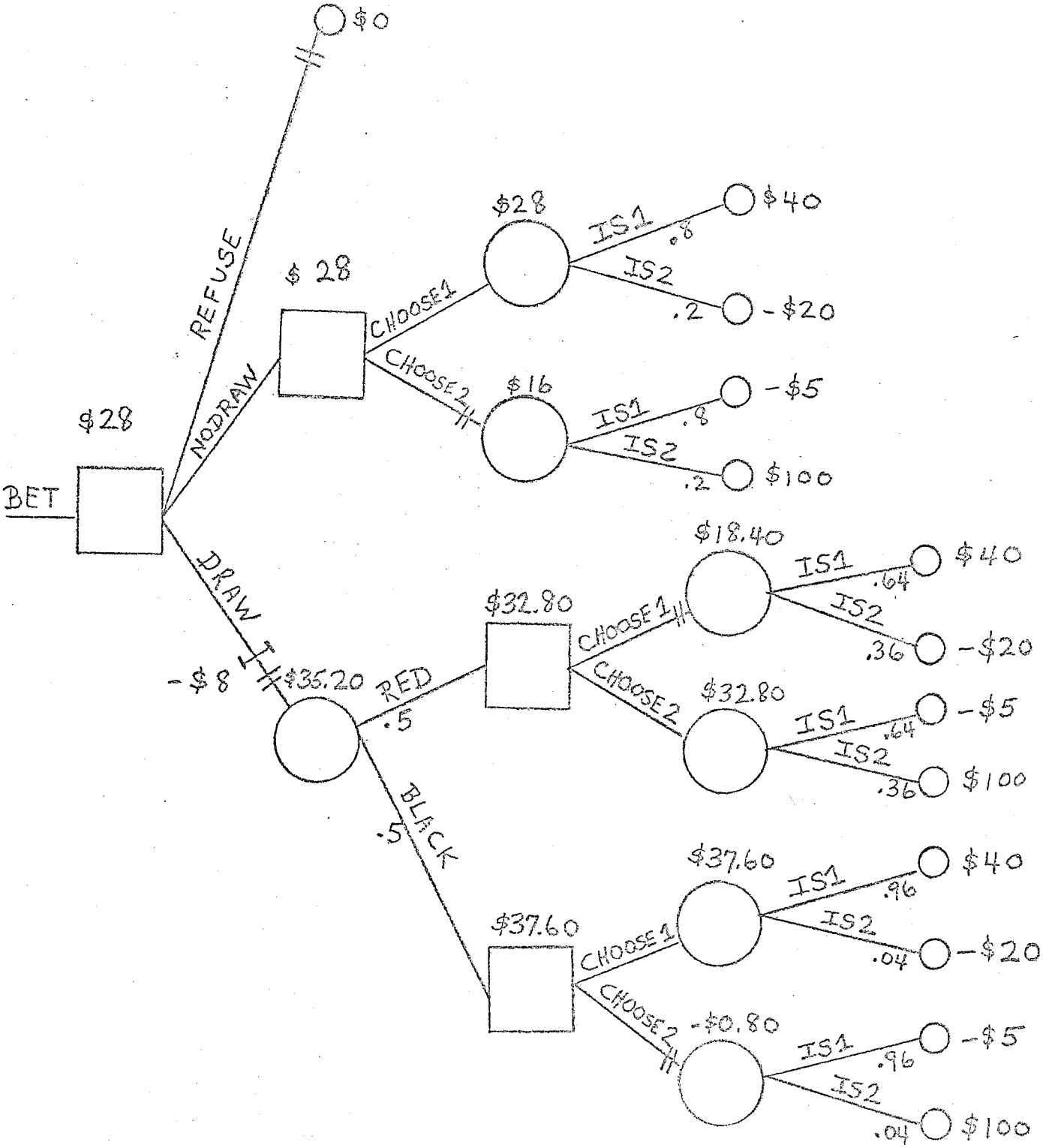

BET
$28
REFUSE
$0
NODRAW
$ 28
CHOOSE 1
$28
IS1
.8
$40
IS2
.2
-$20
CHOOSE 2
$16
IS1
.8
-$5
IS2
.2
$100
DRAW
-$8
$35.20
RED
.5
$32.80
CHOOSE 1
$18.40
IS1
.64
$40
IS2
.36
-$20
CHOOSE 2
$32.80
IS1
.64
-$5
IS2
.36
$100
BLACK
.5
$37.60
CHOOSE 1
$37.60
IS1
.96
$40
IS2
.04
-$20
CHOOSE 2
-$0.80
IS1
.96
-$5
IS2
.04
$100

Figure 2

The computer system described here will carry out the averaging-out-and-folding-back process on a decision tree described to it. One way of presenting the example tree, and the resulting computations, is given below. (Statements beginning with exclamation marks are comments.)

```
? ! LINES BEGINNING WITH AN '!' ARE COMMENTS
? #BET: CHOICE N=#REFUSE,#NODRAW,#DRAW V=0,0,-8
? ! BET IS  A CHOICE POINT WITH BRANCHES NAMES REFUSE,  NODRAW, AND
? ! DRAW, WHICH HAVE VALUES OF 0,0, and -8 RESPECTIVELY.
? ! NEW BRANCH NAMES ARE PREFIXED WITH '#', AND COSTS ARE SHOWN
? ! AS NEGATIVE VALUES.
? NODRAW: CHOICE N=#CHOOSE1,#CHOOSE2
? ! NODRAW WITHOUT A '#' REFERS  TO THE PREVIOUSLY DEFINED NODRAW
? CHOOSE1: CHANCE N=#IS1,#IS2 V=40,-20 W=.8,.2
? ! THE RELATIVE WEIGHTS OF IS1 AND IS2 ARE .8,.2
? CHOOSE2: SAME AS CHOOSE1 V =  -5,100
? DRAW: CHANCE N=#RED,#BLACK W=.5,.5
? RED: SAME AS NODRAW
? .RED.IS1 W=.64
? .RED.IS2 W=.36
? ! .RED.IS1 REFERS TO ALL BRANCHES "FURTHER OUT" THAN RED WITH NAME IS1
? BLACK: SAME AS NODRAW
? .BLACK.IS1  W=.96
? .BLACK.IS2 W=.04
? ! WE MAY NOW DISPLAY THE TREE (TO ANY  DEPTH) AND ALSO AVERAGE
? ! OUT AND FOLD BACK
? DISPLAY TREE BET TO 5
```

BET
?,?,?

REFUSE
!, 0,?

NODRAW
!, 0,?

CHOOSE1
!,?,?

IS1
.8,40,?

IS2
.2,-20,?

CHOOSE2
!,?,?

IS1
.8,-5,?

IS2
.2,100,?

DRAW
!,-8,?

RED
.5,?,?

CHOOSE1
!,?,?

IS1
.64,40,?

IS2
.36,-20,?

CHOOSE2
!,?,?

IS1
.64,-5,?

IS2
.36,100,?

BLACK
.5,?,?

CHOOSE1
!,?,?

IS1
.96,40,?

IS2
.04,-20,?

CHOOSE2
!.?,?

IS1
.96,-5,?

IS2
.04,100,?

```
? ! IN THE TRIPLE GIVEN WITH EACH BRANCH, THE FIRST
? ! IS A '!' FOR A CHOICE PT AND THE APPROPRIATE WEIGHT
? ! (OR A '?', IF UNDEFINED) FOR A CHANCE BRANCH.  THE SECOND
? ! IS COST OR VALUE, IF DEFINED, AND THE THIRD FIGURE IS
? ! THE EXPECTED PAYOFF COMPUTED FOR THAT BRANCH

? AFB BET

EV= 28
OPTIMAL CHOICE IS NODRAW

? DISPLAY TREE BET TO 5

BET
?,?,28
----

REFUSE
!, 0, 0

NODRAW                CHOOSE1               IS1
!, 0,28               !,?,28                .8,40, 0

                                            IS2
                                            .2,-20, 0

                      CHOOSE2               IS1
                      !,?,16                .8,-5, 0

                                            IS2
                                            .2,100, 0

DRAW                  RED                   CHOOSE1               IS1
!,-8,35.2             .5,?,32.8             !,?,18.4              .64,40,0

                                                                  IS2
                                                                  .36,-20, 0

                                            CHOOSE2               IS1
                                            !,?,32.8              .64,-5, 0

                                                                  IS2
                                                                  .36,100, 0

                      BLACK                 CHOOSE1               IS1
                      .5,?,37.6             !,?,37.6              .96,40, 0
```

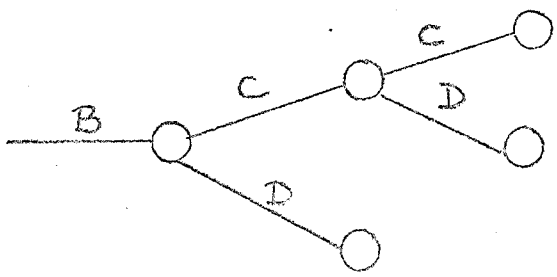
```
                                    IS2
                                    .04,-20, 0

CHOOSE2                             IS1
!,?,-.8                             .96,-5, 0

                                    IS2
                                    .04,100, 0
```

3. THE DECISION TREE LANGUAGE

A formal description of the decision tree language appears in appendix A; this section presents the language informally.

Names refer both to a branch and to the node ending that branch. Simple names are composed of one or more alphabetic characters; as, "X", "BCD", or "ALPHA".

It is possible for a tree to contain several branches with the same name (see below). Compound names are formed by concatenating (with a dot) branch names so as to uniquely specify a single branch. Not all branches leading to a specific branch need be given, nor need those given be adjacent, as long as a unique branch is specified. In case of an ambiguity, the branch closest to the root of the tree is taken; if there is no one closest branch, an ambiguity error occurs. Thus, in the example, C refers to the first (leftmost) branch so named and C.C to the second. C.D and B.D refer to different branches.



Since several branches may have the same name, it is necessary in specifying a branch to indicate whether the name refers to a previously defined branch with that name or a new branch with that name. By convention, branches being newly created are named with simple names preceeded by a "#". Thus, the example tree given above would be defined by specifying:

branch #B with subbranches #C, #D, and then
branch C with subbranches #C, #D.

Occasionally it is desirable to refer to all of the branches with a certain name. This is done by preceding the name with a ".". For example, in the program in section 2, .RED.IS1 refers to two branches, specifically BET.DRAW.RED.CHOOSE1.IS1 and BET.DRAW.RED.CHOOSE2.IS1.

The define command is used to create a new branch and subbranches, to join together existing trees, or to modify existing trees. It is begun by naming a (possibly new) branch. A value, weight (if it emanates from a chance node) and/or a new name may be given for the branch. Values are specified by "V=", weights by "W=", and names by "N=".

Thus, the name of the existing branch now named DRILL is changed to EXPLORE by

DRILL N=EXPLORE ,

and a new branch ALPHA is created with weight 27.3 and value -50 by

#ALPHA V=-50 W=27.3

Substructure for the branch is specified by giving a colon optionally followed by CHOICE or CHANCE to indicate the type of branch and by corresponding lists of subbranch names, values, and (if a chance branch) weights. Alternatively, the substructure can be indicated by indicating a similar branch (e.g., SAME AS or COPY OF) followed by those names, values, and weights which are to be different.

Thus, the example tree given above would be defined precisely with

#B: CHANCE N=#C,#D and
C: CHANCE N=#C,#D .

Weights may be non-negative expressions, or -, meaning the previously defined weight, or ?, meaning undefined. Values may be expressions, -, or ? Expressions are composed of variables or numbers combined using the arithmetic operators +, -, *, /, ↑, and parentheses for grouping variable names are formed using the same rules as simple names.

The assign command assigns the value of the given expression to the specified variable. Thus,

```
GROSS = 3.6
NET = 2.5*GROSS + 1.2↑2
```

assigns the value 3.6 to GROSS and 10.44 to NET.

The display command is used to print out specified information. This may be the value of a variable, the roots branch nodes of all defined trees, a particular tree to a specified depth, or the expected value of a node. In the latter case, if averaging-out-and-folding-back has not yet occurred, a message to that effect will be printed. The four options of display are illustrated by the following.

```
DISPLAY  NET
DISPLAY  ALL ROOTS
DISPLAY  TREE B TO 3
DISPLAY  EV OF B.C
```

The erase command deletes the branch (or branches, if ambiguous) specified and all of its (their) subtrees, as in

```
ERASE  B.C  .
```

The break command disconnects the specified (unique) branch and its subtrees, thus forming a new tree (unless the specified branch was already a root).

```
BREAK  B.D  .
```

The average-out-and-fold-back command causes the specified tree or subtree to be evaluated, defining expected values for each node from the leaves of the tree to the root. The expected value of the specified root and the optimal choice, if a choice node, are printed. For example,

```
AFB  DRILL
```

4. EXTENDING THE EXAMPLE

Suppose that, in the situation described in section 2, you are offered an additional course of action. For a mere $12, you can draw not one but two balls from the selected urn before guessing its type. Should you invest the $12 rather than guessing without any information?

We can construct the tree representing this new alternative with the

following commands.

```
! FIRST, CØMPUTE THE PRØBABILITES ØF DRAWING TWØ
! REDS, ØNE RED AND ØNE BLACK, ØR TWØ BLACKS
PRR = .8*.4*3/9+.2*.9*8/9
PBB = .8*.6*5/9+.2*.1*0
PRB = 1-PRR-PBB
! NEXT, SPECIFY THE TREE
#DRAW2  V=-12:  CHANCE  N=#RR,#RB,#BB  W=PRR,PRB,PBB
RR:  CØPY ØF NØDRAW
.RR.IS1  W= (.4*3/9)*.8/PRR
.RR.IS2  W= (.9*8/9)*.2/PRR
RB:  CØPY ØF NØDRAW
.RB.IS1  W= (.4*6/9+.6*4/9)*.8/PRB
.RB.IS2  W= (.9*1/9+.1*1)*.2/PRB
BB:  CØPY ØF NØDRAW
.BB.IS1  W= 1
.BB.IS2  W= 0
! NØW WE CØULD FIND THE EXPECTED VALUE ØF
! DRAW2 DIRECTLY WITH THE FØLLØWING CØMMAND
AFB  DRAW2
! (ØUTPUT HAS BEEN SUPPRESSED)
! ØR WE CØULD ADD DRAW2 INTØ THE TØTAL
! DECISIØN TREE AND REEVALUATE THAT
BET:  N=DRAW2
AFB  BET
```

Checking that the above commands do perform as specified, and do produce the same result as given in Raiffa, is left as an exercise for the user.

5. FUTURE EXTENSIONS

Two types of additional features can be added to this program, those simplifying the handling of calculations already introduced and those introducing new functions.

Current abilities could be enhanced by introducing notation for (a) referring to branch weights, (b) using Bayes' Theorem, and (c) "flipping"

decision trees. We shall discuss each of these in turn.

(a) Often branch weights can be specified in terms of other branch weights. For example, the weight of branch RR in the example could be given as the weight of R.IS1*3/9 plus the weight of R.IS2*8/9. A more compact notation would be:

W(RR) = W(R.IS1)*3/9 + W(R.IS2)*8/9.

(b) A similar extension would allow the specification of (possibly conditional) probabilities as a special class of variables, and the automatic calculation of a conditional probability using Bayes' Theorem if that probability were undefined but its defining data were given. Thus, in the example, if

P(RR) = W(RR)

P(1) = .8

P(RR|1) = .4*3/9 ,

then

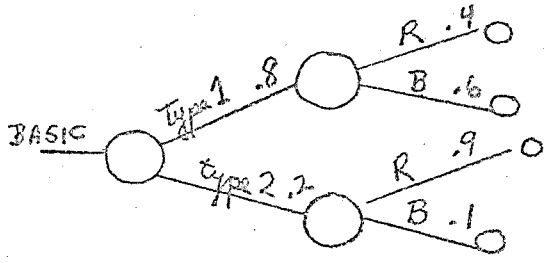
W(.RR.IS1) = P(1|RR)

could result in the automatic calculation of P(1|RR) as

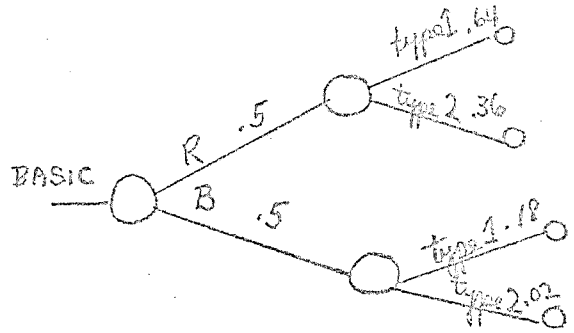
P(RR|1)*P(1)/P(RR).

Since the notation makes clear the interdependencies among probabilities, changes in the probability of an event could result in "automatic" undefining of conditional probabilities involving that event.

(c) In some cases it is easier to find the probabilities of branches in a decision tree by first specifying a related tree and then reordering ("flipping") the tree. For example, the tree



follows directly from the problem statement. Since path probabilities must remain the same, this gives an easy way to compute branch probabilities in the "flipped" tree.



This operation could be carried out with a command of the form
FLIP BASIC .

Among the most useful new functions would be ones enabling specification of (d) individual utility functions, (e) individual subjective probabilities, and (f) normal form of analysis.

(d) For those users of the decision tree program who are unwilling to use the average values of the costs and benefits involved, the program could elicit an estimate of their indifference curve for money by proposing a series of lotteries related to the decision problem. For example, if the decision situation offered a maximum possible gain of $100 and a maximum possible loss of $20, the first lottery would attempt to ascertain the value of an even chance to win $100 or lose $20. Suppose such an opportunity were worth a sure $25 to the decision-maker. The next lottery proposed would be a 50-50 chance at winning $100 or $25, and so forth. The program could generate lotteries and check for "reasonableness" (risk-aversion) in the indifference curve it developed. (Not that risk-preference would be forbidden, but it would be questioned.)

The dialogue establishing an indifference curve might start off as follows.

WOULD YOU RATHER HAVE $40 OR A 50-50 CHANCE AT WINNING
$100 OR LOSING $20?
$40
WHAT ABOUT $30 OR THE 50-50 CHANCE?
$30
$20 OR THE 50-50 CHANCE?
THE BET
FOR WHAT AMOUNT (BETWEEN $30 AND $20) WOULD YOU BE
INDIFFERENT?
$25
OKAY, A 50-50 CHANCE AT $100 OR -$20 IS WORTH $25
TO YOU. NOW LET'S TRY ANOTHER
WOULD YOU RATHER HAVE $65 OR A 50-50 CHANCE AT
WINNING $100 OR WINNING $25?

THE 50-50 CHANCE
THAT SEEMS INCONSISTENT WITH YOUR PREVIOUS ANSWER.

And so forth.

(e) The estimation of subjective (or judgmental) probabilities could follow much the same lottery format to establish the decision-maker's subjective judgment as to the likely outcome of a chance event.

(f) An alternative form of analysis for such decision problems as are discussed here is to spell out all possible strategies, evaluate them for each possible "state of nature" and choose that which maximizes expected return given the decision-makers indifference function for money and judgment as to nature. In our example, such strategies would be: "don't play"; or, "always choose type 1"; or, "always choose type 2"; or, "draw one ball, if red choose type 2, if black choose type 1"; and so forth. Of course, a small subset of the possible strategies dominate all others, and it is from these that the optimum strategy must be chosen. For example, the complete set of strategies for our example problem includes "always choose type 1" and "draw one ball, if red choose type 1, if black choose type 1". Clearly, no matter what the probability of type 1, the latter strategy returns $8 less than the former.

Given the decision tree in the form specified earlier, it would be straightforward for a mechanical procedure (a program) to select the dominant strategies and list them according to the probabilities of various states for which they were optimum.

APPENDIX A -- FORMAL SYNTAX

<command> ::= <define> | <assign> | <display> | <erase> | <break> | <afb> | <comment> *

<display> ::= DISPLAY <display spec> **

<display spec> ::= TREE <compound name> TO <integer> | ALL ROOTS | EV OF <compound name> | <expression>

<erase> ::= ERASE <dotted name>

<break> ::= BREAK <compound name>

<afb> ::= AFB <compound name>

<assign> ::= <simple name> = <expression>

<comment> ::= ! <anystring>

<define> ::= <branch spec>.<NVW spec> |
<branch spec> <NVW spec> : <structure spec>

<branch spec> ::= <branch name> | <dotted name>

<NVW spec> ::= [N = <simple name> / W = <weight> / V = <value>] ***

<weight> ::= <expression> | - | ?

<value> ::= <expression> | - | ?

<structure spec> ::= <copy spec> <wvl spec> | <type> <nwvl spec> | <wvl spec>

<copy spec> ::= COPY OF <compound name> | SAME AS <compound name>

<type> ::= CHOICE | CHANCE

* meaning: a command is a define or a assign or a display or ...

** meaning: a display is the word DISPLAY followed by a display spec.

*** meaning: a NVW spec is any or all of the alternatives, in any order, but without repetitions.

```
<nwvl spec>        ::=  [N = <name list>
                         W = <weight list>
                         V = <value list>]

<wvl spec>         ::=  [W = <weight list>
                         V = <value list>]

<name list>        ::=  <name>,<name list> | <name>

<weight list>      ::=  <weight>,<weight list> | <weight>

<value list>       ::=  <value>,<value list> | <value>

<expression>       ::=  <expression> + <term> | <expression> - <term> |
                        + <term> | - <term> | <term>

<term>             ::=  <term> * <factor> | <term> / <factor> | <factor>

<factor>           ::=  <factor> ↑ <primary> | <primary>

<primary>          ::=  <simple name> | <number> | (<expression>)

<dotted name>      ::=  .<compound name> | <compound name>

<compound name>    ::=  <simple name> . <compound name> |<simple name>

<branch name>      ::=  # <simple name> | <simple name>

<simple name>      ::=  <character> <simple name> | <character>

<number>           ::=  <digits> . <digits> | . <digits> | <digits> . | <digits>

<integer>          ::=  <digits>

<digits>           ::=  <digit> <digits> | <digit>
```

PRIMITIVES:

```
<character>     A|B|....|Z
<digit>         0|1|... |9
<anystring>
```

APPENDIX B -- TYMSHARE IMPLEMENTATION

The language discribed in this paper has been implemented on the interactive computing service, TYMSHARE; this implementation was coded in TYMSHARE's SUPERBASIC. While exact details of access to that system are to be found elsewhere, the following comments concerning the implementation are of more general interest.

1) Two additional decision language commands allow the user to save and recall the current status of defined trees and variables. These commands are SAVE and RECALL. The system responds to SAVE by requesting FILENAME?; any name may be supplied. That same name, say DECISION, is used in recalling the file at a later time, as in RECALL DECISION.
2) Errors in the form (syntax) of a command result in an error number and a partial repetition of the input line indicating where the apparent error occurred. Other errors during processing result in error messages.
3) The maximum number of branches that can be defined at any time is 100.