Adding Probabilistic Decision Making to EXPERT 2

L. Glen Watson
Department of Mechanical Engineering
University of Saskatchewan, Saskatoon, Saskatchewan, S7N 0W0

In this paper, the author discusses a method of extending EXPERT2 [1], the well known FORTH expert system shell, by incorporating a structure to assign Bayesian probabilities to the hypothesis in the knowledge base and then by using RUNWORDS to manipulate the probabilities after proving any THEN. After the acceptance of a hypothesis, the probabilities of each of the hypothesis being true is listed. The manipulation of the probabilities is carried out as suggested by Charniak and McDermott [2].

Charniak and McDermott, argue that even though the probabilities of occurrence of various symptoms are not independent of each other, useful information can be gained by assuming they are and manipulating the conditional probabilities accordingly. This method of handling the probabilites in EXPERT2 should be of interest. The method used for adding the probability handling to EXPERT2 can also be used for assigning fuzzy confidence numbers [3] to the EXPERT2 decisions.

The Implementation

The code shown was developed on and for the most part used on the LMI 32 bit Forth implementation Forth+. It has also been used with two 32 bit forth implementations on the ATARI 1040. The only adjustment was to shorten the length allotted for floating point number storage. The screens listed below contain all of the information necessary for the probabilistic additions.

Screen # 2

```
( PROB structure 2                      LGW 11:09 04/25/87 )
 : PROB ( n --- )
       CREATE  64 \ THENHYP string space
               8  \ space for a real probability
               10 \ space for usage list
       + + * ALLOT DOES> ;
: PUTL ( n addr ---- ) \ put n in first available slot
       DUP C@ IF  1+ RECURSE ELSE C! THEN ;
: GETL ( addr --- )   \ read and print usage list
 DUP    10 + SWAP DO I C@ DUP IF . ELSE DROP SPACE THEN LOOP ;
   (   DUP C@ DUP IF . 1+ RECURSE ELSE DROP THEN ; )
```

Screen 2 contains a CREATE-DOES> word PROB to set up a structure for the probabilities for each hypothesis. The record asociated with each hypothesis includes a descriptive string of 64 characters, the probability, and a 10 byte space for a list to record the number associated with the rules which manipulated the probability associated with that hypothesis. The PUTL is used to place the usage numbers in the PROB structure and GETL is used to print out the numbers stored in the list.

Screen # 3

```
( PROB structure 3 initialization        LGW 10:50 04/28/87 )
 8 PROB ZOO  \ Set up prob structure for the zoo
 VARIABLE INTEGER FVARIABLE REAL   \ variables for FISWAP
FVARIABLE INIT-PROB 1.0E 7.0E F/ INIT-PROB F!
 : FISWAP ( n f -- f n)  \  exchange a real and an integer
    REAL F! INTEGER ! REAL F@ INTEGER @ ;
 : string-store SWAP OVER C@ 1 + CMOVE ;
 : prob!  DUP 64 + INIT-PROB F@ FISWAP F! ;
 : list-clear 72 + 10 0 FILL ;
 : animal-init string-store prob! list-clear ;
 : CHEETAH-INIT       ( addr --- )
       DUP  " animal is cheetah" \ store the counted string
       animal-init ;
```

Loading screen 3 sets up the structure ZOO which in this case is set up for only 8 hypotheses. The integer variable INTEGER and the floating point variable REAL are set up to use in the word FISWAP. FISWAP is used to swap the floating point number at the top of the stack with an integer located just under it. In FORTH+ the word FISWAP is just ROT, while other FORTH implementations will be different depending upon the size of the floating point and integer words and whether or not there is a separate floating point stack. INIT-PROB contains the initial probability for each hypothesis. If the initial probilities are not the same this code will have to be changed to assign different values to each hypothesis. The word string-store is used to store a string associated with the hypotheses into the PROB structure, while the word prob! inserts the initial probability of each hypothesis into its associated slot and the word list-clear sets each cell of the usage list to ASCII 0. Next animal-init uses each of the three words above to initialize the record for some animal. Finally CHEETAH-INIT illustrates how these words are used.

Screen # 6

```
( PROB structure 6 initialization        LGW 10:56 04/24/87 )

 : ZOO-INIT \ initialize the ZOO PROB structure
       ZOO DUP CHEETAH-INIT 82 + DUP  TIGER-INIT
     82 + DUP  ZEBRA-INIT   82 + DUP  GIRAFFE-INIT
     82 + DUP  OSTRICH-INIT 82 + DUP  PENGUIN-INIT
     82 +      ALBATROSS-INIT ;
```

Screen # 7

```
( PROB structure 7 structure print        LGW 10:53 04/28/87 )
: .ZOO  \ Print out contents of the PROB structure ZOO
      ZOO CR  UNDERLINE ." Probability table "
      -UNDERLINE CR INTENSITY
      CR ." Hypothesis " 40 OUT @ - SPACES ." probability"
      ."                rules " -INTENSITY
       7 0 DO    CR DUP   DUP COUNT TYPE 40 OUT @ -  SPACES
              DUP 64 + F@ 8 FP.R 3 SPACES    72 + GETL 82 +
           LOOP DROP 1 ;
:  prob_update  FLOG F+ FALOG FISWAP F!  ;
: prob_get
       82 + DUP DUP F@ FLOG ;
```

```
Screen # 8
( PROB WORDS IsBird3                             LGW 12:04 04/24/87 )
: 3PUTL DUP 8 + 3 SWAP PUTL ;
: IsBird3   ZOO 64 + DUP  DUP
        F@ FLOG  0.00004E 7E F/ prob_update  3PUTL
        prob_get 0.00004E 7E F/ prob_update  3PUTL
        prob_get 0.00004E 7E F/ prob_update  3PUTL
        prob_get 0.00004E 7E F/ prob_update  3PUTL
        prob_get        7E 3E F/ prob_update  3PUTL
        prob_get        7E 3E F/ prob_update  3PUTL
        prob_get        7E 3E F/ prob_update  3PUTL
        DROP TRUE ;
```

Screen 6 is merely to define a word to initialize the whole ZOO structure.

Screen 7 contains a word .ZOO which is used to print out the contents of ZOO and two other words prob_update and prob_get which are used by the runwords to update the probabilities in ZOO.

Screen 8 shows the way the PUTL is used in the runword IsBird3, which appears in screen 22 as a runword in rule 3. The final computer output is the result of a typical session with this enhanced version of EXPERT2.

```
I DEDUCE
  animal is probably cheetah

I DEDUCE
  animal is cheetah

Probability table

Hypothesis                          probability              rules
animal is cheetah                   1.000000         2 11
animal is tiger                     0.187500         2 11
animal is zebra                     0.187500         2 11
animal is giraffe                   0.187500         2 11
animal is ostrich                   0.061224         2
animal is penguin                   0.061224         2
animal is albatross                 0.061224         2
I CONCLUDE
  animal is cheetah
ok
```

DISCUSSION

The display of the probabilities at the end of the output can be useful for those cases where there is not a clear cut answer to the problem. This usually shows up with two or three hypotheses having very similar probabilites of truth. The person using an expert system with the enhancement will probably appreciate it. Unfortunately, the work required to produce the expert system is considerably greater as all of the probabilities have to be calculated or at least estimated. The good part of this is that it forces you to get a good understanding of the problem just to keep the output of the probabilities from looking ridiculous.

According to L.A. Zadeh [4], the guru of fuzzy set theory and fuzzy arithmetic, clients are much happier with expert systems that give positive answer or overly optimistic probabilities, so this enhancement may not be good for business. Dr. Zadeh also expressed the opinion that fuzzy possibilities should be used for expert systems and not probabilities. It appears that the extension to EXPERT2 outlined in this paper can be easily adapted to possibilities.

REFERENCES

1. Park, J., "A consequent-Reasoning Inference Engine for Microcomputers" 1984 FORML Conference Proceedings, November 23-24, Asilomar Conference Center Pacific Grove, California.

2. Charniak, E. and McDermott, D., "Introduction to Artificial Intelligence" Addison-Wesley Publishing Company, 1984.

3. Gupta, et al, "Approximate Reasoning in Expert Systems", North Holland, 1985.

4. Zadeh, L.A., "Lecture at the University of Saskatchewan", April 6, 1987.