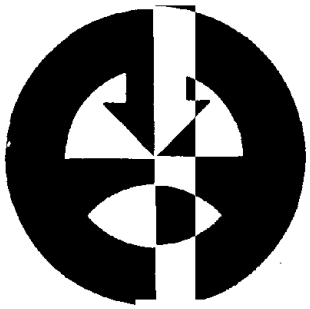
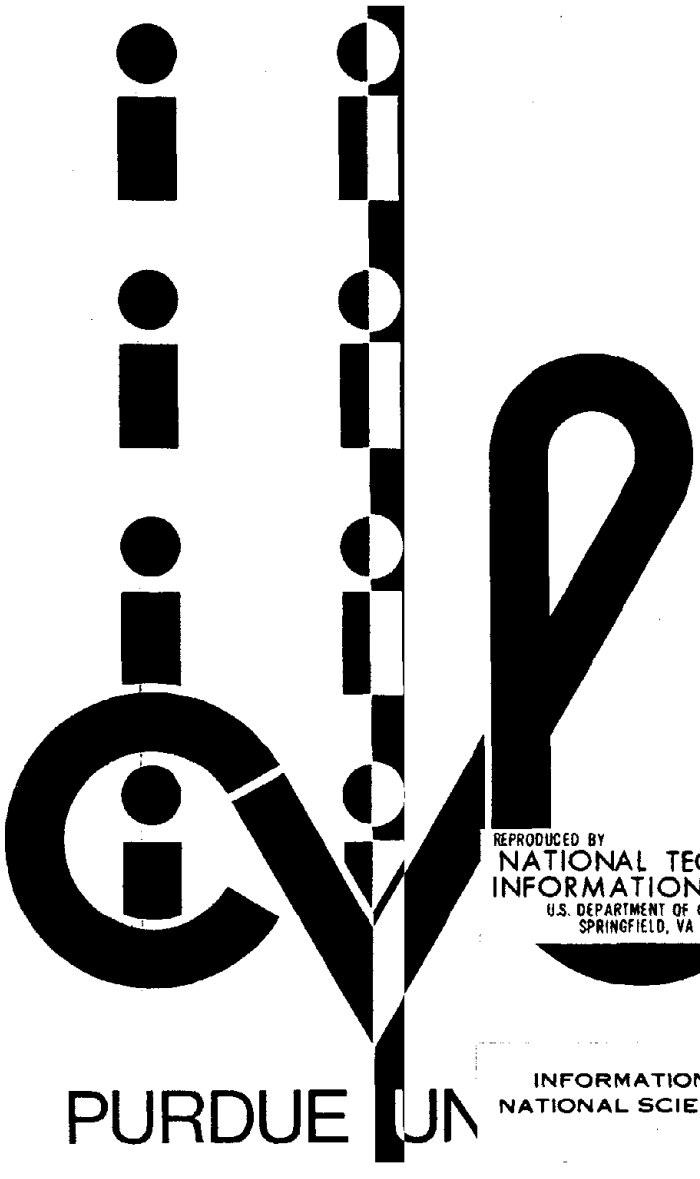
SCHOOL OF CIVIL ENGINEE

NSF/CEE-81075

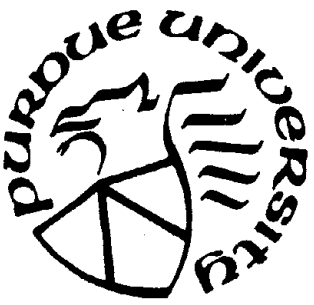
PB82-158460



# Structural Engineering








PURDUE UN

INFORMATION RESOURCES
NATIONAL SCIENCE FOUNDATION

CE-STR-81-36

# SPERIL I - COMPUTER BASED STRUCTURAL DAMAGE ASSESSMENT SYSTEM

M. Ishizuka
K. S. Fu
J. T. P. Yao



Supported by
The National Science Foundation
through
Grant No. PFR 796296

November 1981

School of Civil Engineering
Purdue University
West Lafayette, IN 47907

# SPERIL I - Computer-Based Structural Damage Assessment System

by M. Ishizuka[1], K. S. Fu[1], and J. T. P. Yao[1]

## 1. Overview

SPERIL is a computer-based damage assessment system of existing structures, which are subjected to earthquake excitation. Its objective and underlying theory have been already described elsewhere [1-4]. To obtain an appropriate answer which is, in this case, the classification of damage state, SPERIL can be used to interprete observation data including the results of analyzing accelerometer records and on-site visual inspection.

The expert system approach, which allows us to decompose a complex decision-making problem into several simpler hierarchical subproblems and to collect related useful knowledge as fragmentary rules, has been employed to achieve highly effective utilization of experienced engineers' knowledge. This approach can be used to provide the large capability of dealing with a great variety of structural conditions involved in the structural damage assessment problem. Because the related knowledge often contains uncertainty and impreciseness of expression range, a new inference procedure with uncertainty and fuzzy restriction has been developed. The inference proceeds to obtain certainty measures at higher subgoals by using available rules and observed data, and eventually gives certainty measures at final goal state, which suggest an appropriate answer. In SPERIL version I, separate evidential observations are integrated on the basis of Dempster & Shafer's theory for fuzzy subset.

In this report, the implementation of SPERIL program (version I) is outlined. It is noted that this program is a preliminary version. The rules as written in rule-base are expected to be refined and updated with more accurate and specific rules in subsequent investigations. This first version, however

[1]Visiting Associate Professor of Electrical Engineering, Goss Professor of Engineering, and Professor of Civil Engineering, respectively, Purdue Univ., West Lafayette, IN 47907.

does show the feasibility of a systematic computer-based damage assessment system. The program portion is written by using language C with some 800 statements. On the other hand, the rules in the rule-base are written in a format close to natural rule sentences.

## 2. Rule Representation

Figure 1 shows the whole program configuration of SPERIL which consists of three spearate files. Rule-base (file name: Rbase) is a completely separate storage from control and inference process. Useful knowledge for the inference purpose has been collected under the organization as shown in Fig. 2 and expressed in a stylized rule format which will be described subsequently.

The rule format is designed as shown in Fig. 3 so that both human and computer can interprete it easily. First line of the rule is Rule and followed with a 4-digit rule number, the first two digits of which are rule set number corresponding to the node numbers as shown in Fig. 2. Following the first line, rule statements are written in as many lines as it is necessary. As shown in Fig. 3, each statement line has headers such as field-1, field-2 and field-3. Eleven types of headers as listed in Table 1 are recognized by the computer. The end of one rule is found by encountering unrecognizable header. The line with header type -18 is premise statement; and the line with header type 9 and 10 is action statement. The fundamental function of production system, that is, "if premise is satisfied then action is taken" is emphasized in rule interpretation. The action in this case is an updating process of short term memory.

In the field-1, the name of short term memory is written. The interpretation of field-2 and field-3 is determined according to the type of header and type of short term memory which will be described in detail in the following section. In the present version of SPERIL, the combinations of Table 2 are defined and implemented for the processing. The interpretations of these statements are straightforward. As fuzzy subsets which represent the fuzzy grade

of damage in this particular case, acceptable expressions are restricted to six, namely, no, slig (slight), mode (moderate), seve (severe), dest (destructive) and uk (unknown). These fuzzy subsets are defined as shown in Fig. 4. If the statement is premise and has "t" in position 21, the following four positions perform as optional field-3, which is used to indicate a union subset. Remaining positions are available for comments.

## 3. Short Term Memory

Short term memories are working memory spaces for inference in which the input data or inferred data are stored. In SPERIL, the following four (in fact 13 because smd is an array) spaces are reserved in main program for each short term memory:

```
 char smn [3]     :name of short term memory
 int  smt         :type of short term memory (1-4)
float smd [10]    :numerical data array
 char sml [4]     :linguistic data
```

(char, int. float are language C definition indicating character, integer, floating point, respectively)

Three characters name (usually upper case) is written in smn[3] to identify the short term memory. Each short term memory is classified into four types (1-4), one of which is indicated in each smt. Table 3 shows those types of short term memory and the meanings of their memory contents. Each character memory sml is initialized to "unan" to serve as an indicator showning that the short term memory has not been written as yet. Whenever the premise is examined or action is taken, the type of short term memory is referred to proceed to an appropriate interpretation of the rule statement.

## 4. Control and Inference

Figures 5 and 6 show the control and rule-based inference flow of SPERIL. Because the inference network is not deep, no heuristic or sophisticated strategy of rule invocation is adopted. The sequence of rule set invocation is pre-assigned as follows:

"05", "06", "07", "08", "09", "10", "02", "03", "04", "01".

This corresponds to bottom-up search rather than top-down or goal-oriented research.

If a relating rule is found in the rule-base, it is processed according to Fig. 6. When the premise is examined and the associated short term memory is found to be unwritten or unanswered, a question is issued to get data. The question is generated referring to a question file (file name: quest) in which an appropriate question sentence is stored for each short term memory with possibility of getting data from operator rather than the inference process.

To avoid the issue of annoying unnecessary questions, skip pass is provided for the case that there is no possibility for later action statements to be taken. Thus only minimum necessary questions are issued for the purpose of inference.

After one rule is processed, the resultant content of buffer memory is used to update the short term memory indicated by field-1 of the action statement. For type-1 short term memories, the Dempster & Shafer's theory extended to fuzzy subset is employed for their updating or combining separate evidences.

Final decision is made according to Dempster & Shafer's lower probabilities of the fuzzy subsets in final goal (FIN) which is damage state. If no fuzzy subset has larger lower probability than a certain threshold (0.2), SPERIL selects no appropriate answer. Therefore, the answer is one of the following:

1) no damage,
2) slight damage,
3) moderate damage,
4) severe damage,
5) destructive damage,
6) no appropriate answer.

SPERIL version I is currently working on a PDP11/45 which can be accessed on the Purdue EE Unix network.

Acknowledgements

This research program has been supported by the National Science Foundation through Grant No. PFR-796296. We wish to thank Mrs. Vicki Gascho for her excellent

typing of this report.

## References


[1] Ishizuka, M., Fu, K. S. and Yao, J. T. P. "Inexact Inference for Rule-Based Damage Assessment of Existing Structures," Seventh International Joint Conference on Artificial Intelligence, Vancouver, Aug. 1981.

[2] Ishizuka, M., Fu, K. S. and Yao, J. T. P., "A Rule-Inference Method for Damage Assessment," American Society of Civil Engineers Convention, St. Louis, Oct. 1981.

[3] Ishizuka, M., Fu, K. S. and Yao, J. T. P., "Inference Procedure with Uncertainty for Problem Reduction Method," Purdue Univ. Structural Engineering Report, CE-STR-81-24, Aug. 1981, also submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence.

[4] Ishizuka, M., Fu, K. S. and Yao, J. T. P., "A Rule-Based Inference with Fuzzy Set for Structural Damage Assessment," submitted to Gupta, M. M. and Sanchez, E., Ed. Fuzzy Information and Decision Processes, North Holland Publ. Co. 1982.


Table 1. Defined header of rule statement.

| Header | Header Type | |
|---|---|---|
| IF | 1 | |
| OR IF | 2 | |
| THEN IF | 3 | |
| ELSE IF | 4 | |
| THEN | 9 | |
| ELSE | 10 | |
| and | 5 | (in the context of 1.2) |
| | 7 | (in the context of 3.4) |
| | 11 | (in the context of 9) |
| or | 6 | (in the context of 1.2) |
| | 8 | (in the context of 3.4) |

Table 2. Defined rule statement in SPERIL.

| Header | Type of short term memory (field-1) | Field-2 | Field-3 |
|---|---|---|---|
| Premise | 1 | is | (fuzzy subset) no, sligh, mode, seve, dest, uk |
| | 2 | is | linguistic data |
| | 3 | >=, >, <=,< | numerical data or name of short term memory of type-3 |
| | 4 | is | yes, no |
| Action | 1 | (fuzzy subset) no, slig, mode, seve, dest, uk | certainty measure |
| | 2 | = (substitute) | linguistic data |
| | 3 | =, ++ (accumurate) | numerical data |

Table 3. Types of short term memories.

| Type | Meaning of memory contents |
|---|---|
| 1 | Certainty measures of fuzzy subsets<br>no : smd[0]<br>no slig : smd[1]<br>slig : smd[2]<br>slig mode : smd[3]<br>mode : smd[4]<br>mode seve : smd[5]<br>seve : smd[6]<br>seve dest : smd[7]<br>dest : smd[8]<br>uk : smd[9] |
| 2 | Linguistic data (up to 4 characters) : sml |
| 3 | Numerical data : smd[0] |
| 4 | Yes - no data<br>yes : smd[0]=1<br>grey : smd[0]=0.6, smd[1]=0.2, smd[2]-0.2<br>no : smd[1]=1<br>uk : smd[2]=1 |

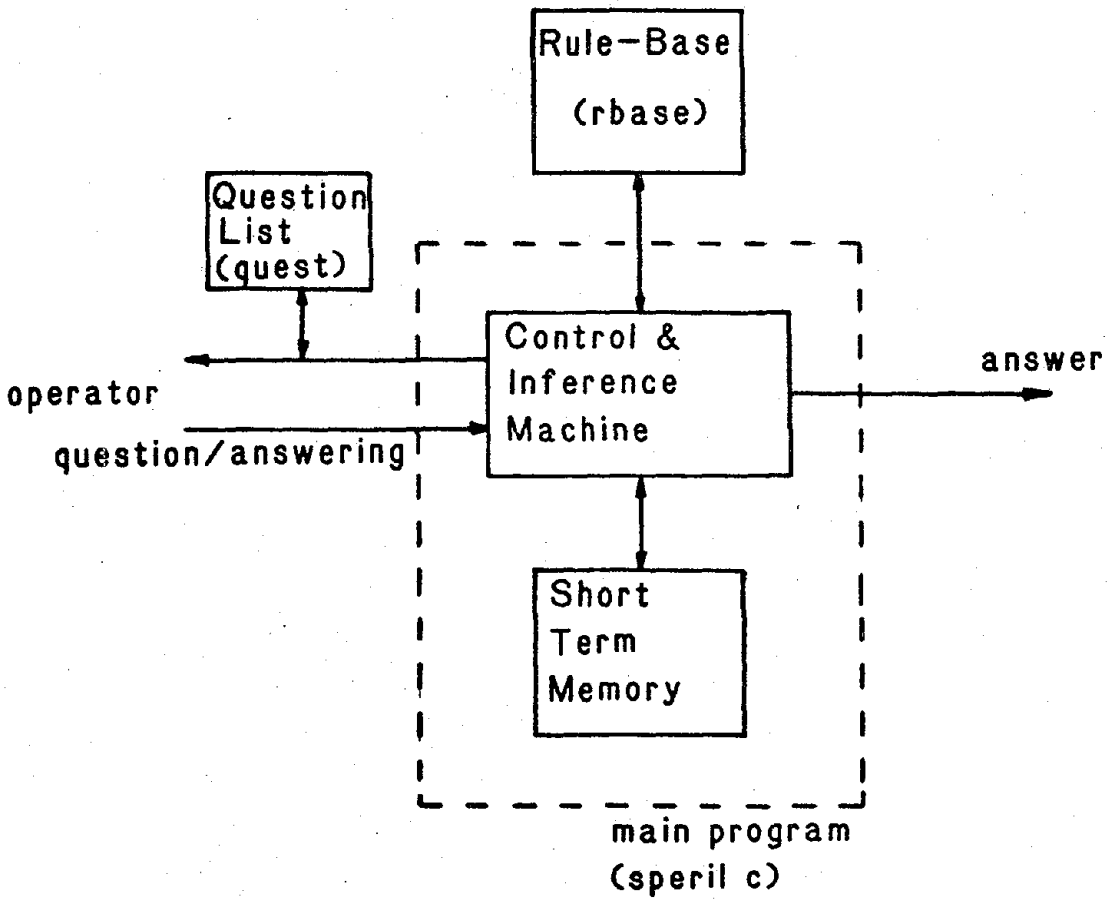



Fig. 1. Configuration of SPERIL program.

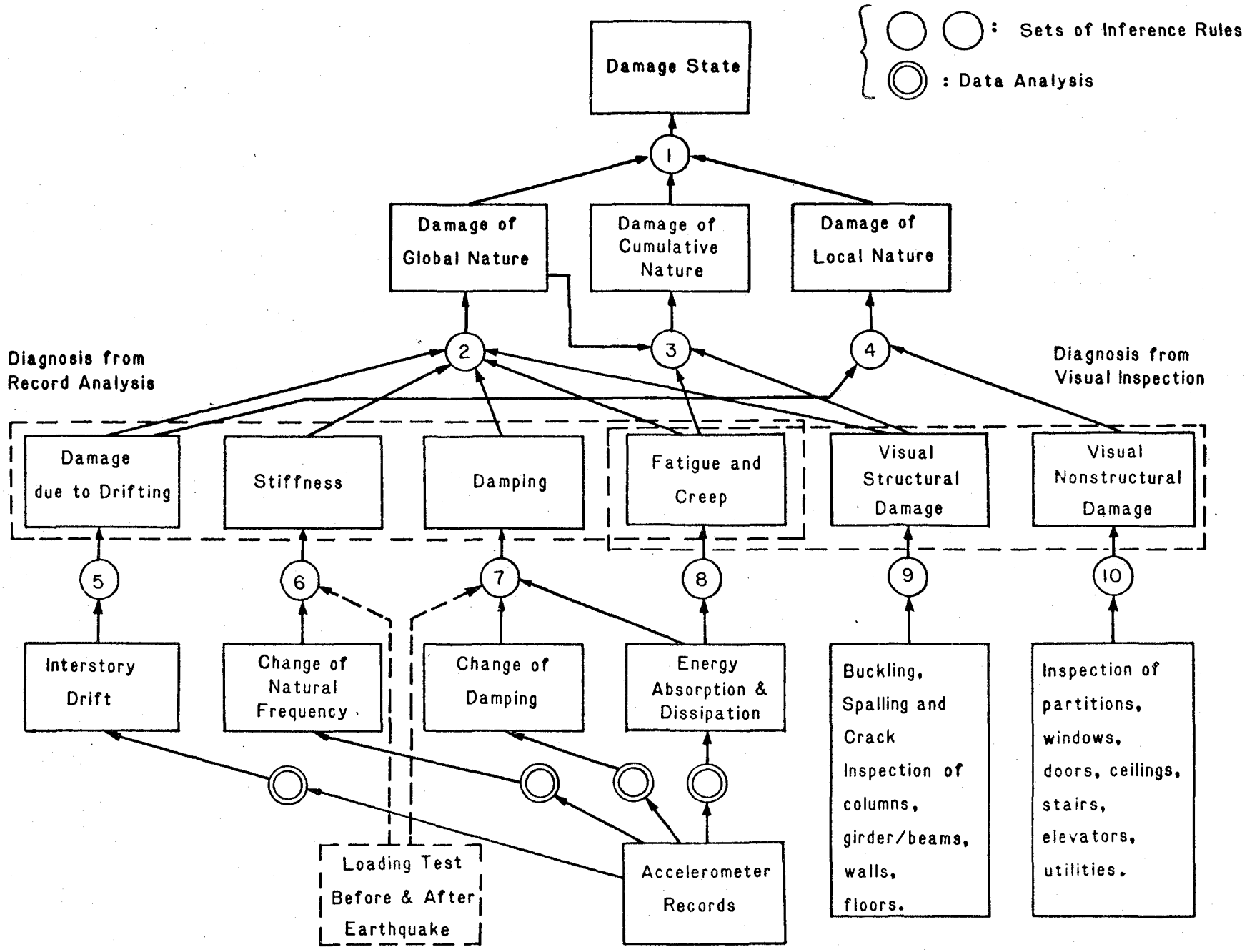



Fig. 2. Inference network of SPERIL.

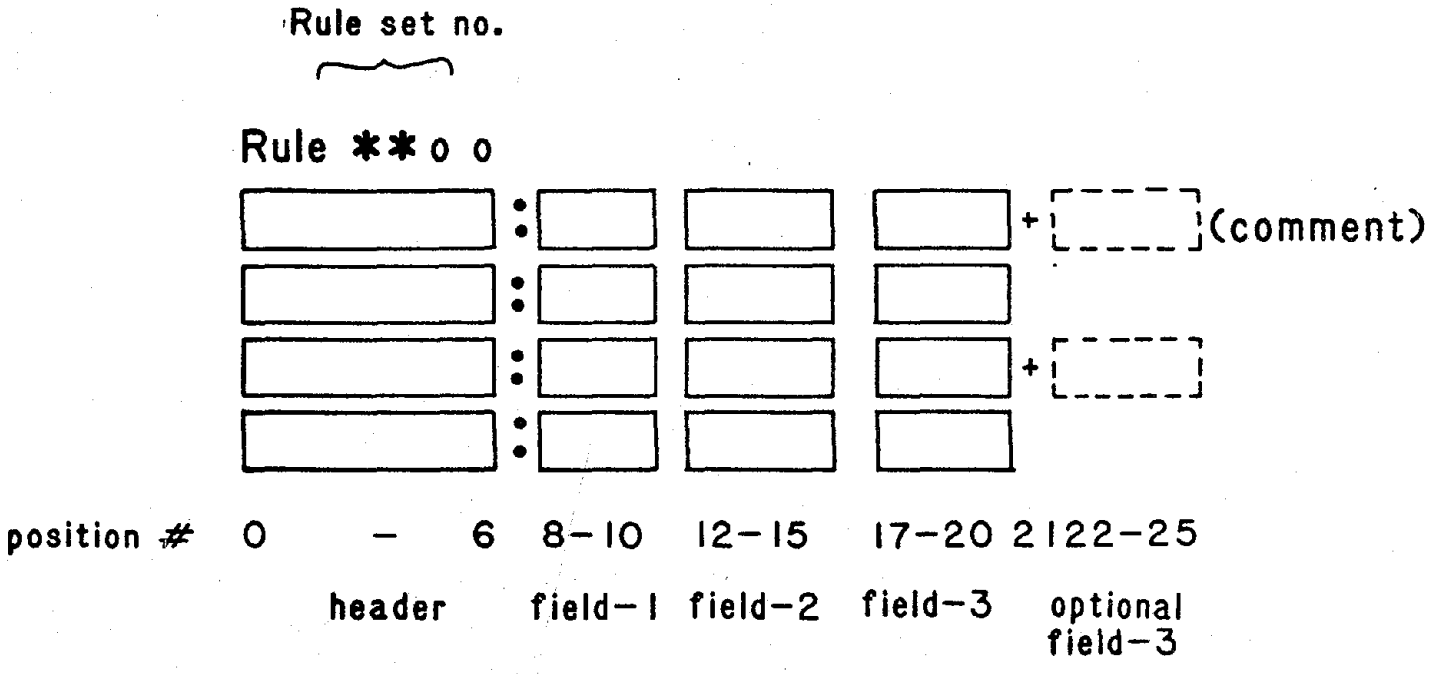



Fig. 3. Rule format of SPERIL.

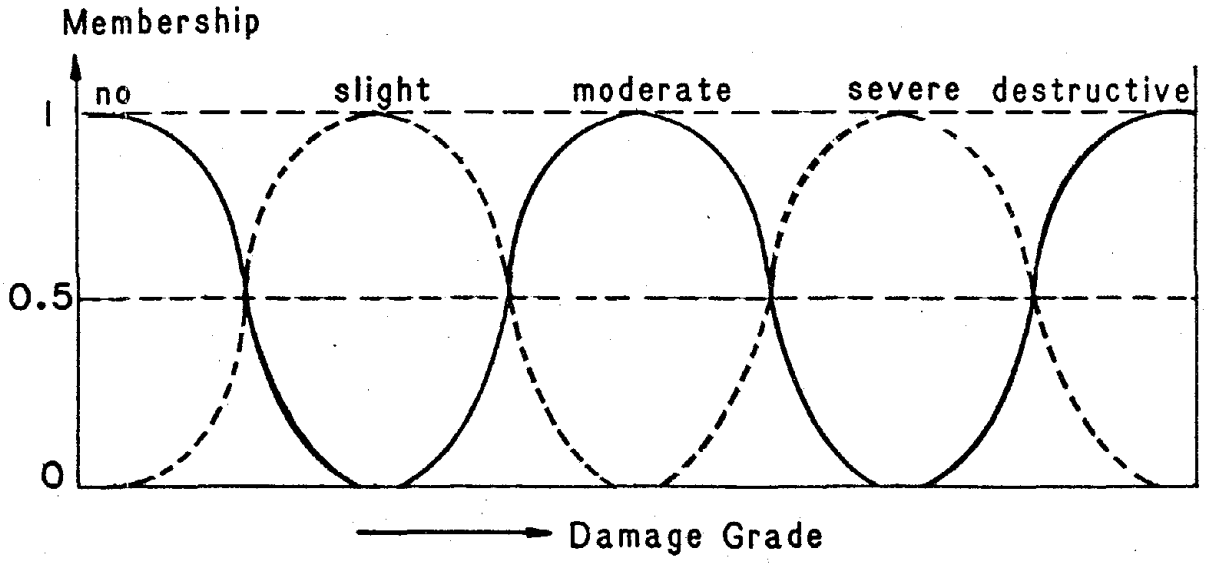



Fig. 4. Fuzzy grade used in SPERIL.

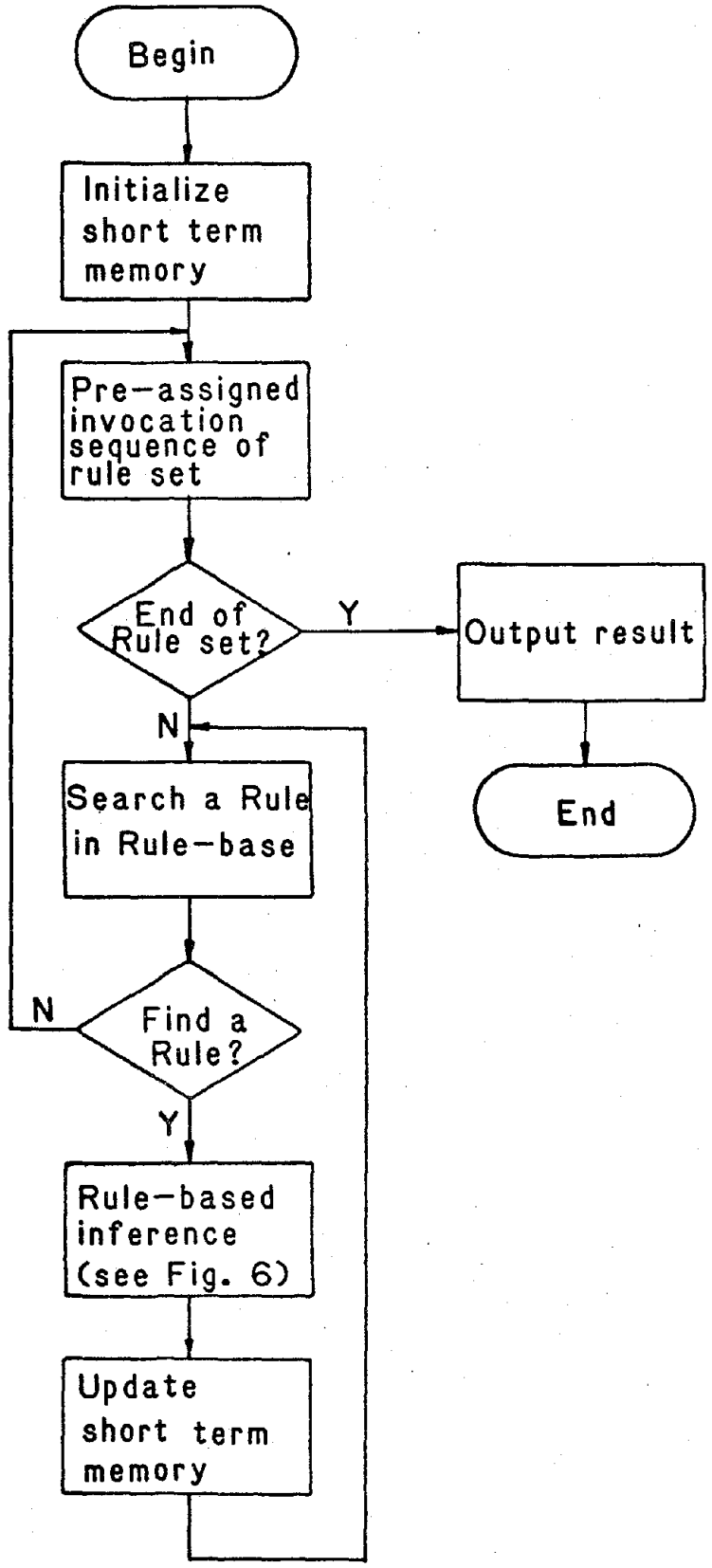



Fig. 5. Main control flow of SPERIL.

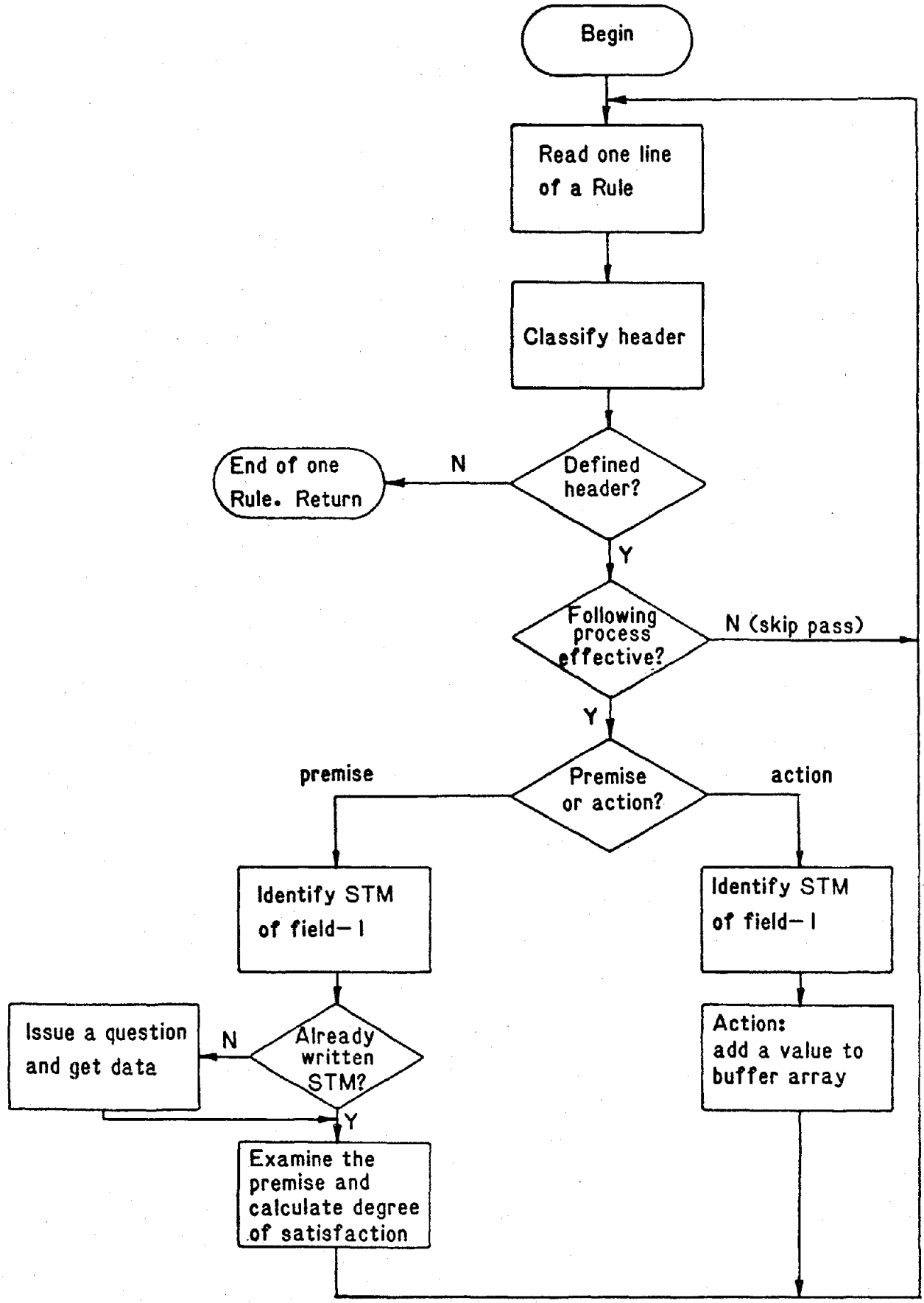



Fig. 6. Processing of one rule.
(STM: short term memory)

APPENDIX A SOURCE PROGRAM

```
1=/****** speril.c     ----    SPERIL (version I) main program ******/
2=/*       with files of rbase(rule-base) and quest(questions)      */
3=
4=#include <stdio.h>
5=
6=  /* short term memory */
7=     /* smt[i]           */
8=     /*  1       smd[0-9]:certainty measures (Dempster & Shafer's
9=                          basic prob.) of no, no-slig, slight,
10=                         slig-mode, moderate, mode-seve, severe,
11=                         seve-dest, destructive, unknown.
12=          2        *sml    :linguistic data.
13=          3        smd[0]  :numerical data.
14=          4       smd[0-2]:yes, no, unknown.   */
15=#define NSTM 50
16=  char  smn[NSTM][4]  ;        /*name*/
17=  int   smt[NSTM];            /*type*/
18=  float smd[NSTM][10];        /*numerical data*/
19=  char  sml[NSTM][5]  ;        /*linguistic data*/
20=  int   pp;                   /*control of print-out*/
21=
22=main()
23={
24=  char bname[50], date[10];
25=  char *seq[15];
26=  char rsno[3];  /*rule set no.*/
27=  int i=0;
28=  extern int pp;
29=
30=  /*sequence of rule set invocation*/
31=  seq[0]="05";
32=  seq[1]="06";
33=  seq[2]="07";
34=  seq[3]="09";
35=  seq[4]="10";
36=  seq[5]="02";
37=  seq[6]="04";
38=  seq[7]="01";
39=  seq[8]="$$";
40=
41=  printf("****************** SPERIL  (Version I) ******************** \n
42=  printf(" Rule-based Damage Assessment System for Existing Structure \n
43=  printf(" subjected to Earthquake Excitation \n");
44=  printf("*************************************************************\n"
45=  printf("Answer the following questions.\n");
46=  printf("Then I can suggest a reasonable answer. \n\n");
47=
48=  printf("Enter the name of building. =");
49=  scanf("%s",bname);
50=  printf("Enter date. = ");
51=  scanf("%s",date);
52=  printf("Do you want to see rule sentences on display (y or n) ? =");
53=  scanf("%s",rsno);  pp=rsno[0];
54=
55=  initial();                          /*initilize*/
56=
57=  /*main loop*/
58=  while (*seq[i] != '$')
59=   { rsno[0] = *seq[i];
60=     rsno[1] = *(seq[i]+1);
```

```
 61=      rsno[2] = *(seq[1]+2);          /* '\0' : end of string*/
 62=
 63=      rbinfer(rsno);                 /****rule-based inference****/
 64=      i++;
 65=    }
 66=
 67=   result();
 68=}
 69=
 70=
 71=
 72=
 73=
 74=/*** initialize short term memory ***/
 75=initial()
 76={
 77=   extern char  *smn[];
 78=   extern int    smt[];
 79=   extern float  smd[][10];
 80=   extern char  *sml[];
 81=   int i, n, j;
 82=
 83=   smn[0]="FIN";   smt[0]=1;   sml[0]="unan";
 84=   smn[1]="GLO";   smt[1]=1;   sml[1]="     ";
 85=   smn[2]="CUM";   smt[2]=1;   sml[2]="     ";
 86=   smn[3]="LOC";   smt[3]=1;   sml[3]="     ";
 87=   smn[4]="STI";   smt[4]=1;   sml[4]="     ";
 88=   smn[5]="DAM";   smt[5]=1;   sml[5]="     ";
 89=   smn[6]="FAT";   smt[6]=1;   sml[6]="     ";
 90=   smn[7]="VST";   smt[7]=1;   sml[7]="     ";
 91=   smn[8]="VNS";   smt[8]=1;   sml[8]="     ";
 92=   smn[9]="MAT";   smt[9]=2;   sml[9]="     ";
 93=   smn[10]="INS";   smt[10]=2;   sml[10]="     ";
 94=   smn[11]="FLX";   smt[11]=2;   sml[11]="     ";
 95=   smn[12]="N01";   smt[12]=2;   sml[12]="     ";
 96=   smn[13]="N02";   smt[13]=2;   sml[13]="     ";
 97=   smn[14]="N03";   smt[14]=2;   sml[14]="     ";
 98=   smn[15]="N04";   smt[15]=2;   sml[15]="     ";
 99=   smn[16]="N05";   smt[16]=2;   sml[16]="     ";
100=   smn[17]="N06";   smt[17]=2;   sml[17]="     ";
101=   smn[18]="N07";   smt[18]=2;   sml[18]="     ";
102=   smn[19]="N08";   smt[19]=2;   sml[19]="     ";
103=   smn[20]="N09";   smt[20]=2;   sml[20]="     ";
104=   smn[21]="N10";   smt[21]=2;   sml[21]="     ";
105=   smn[22]="N11";   smt[22]=2;   sml[22]="     ";
106=   smn[23]="ISD";   smt[23]=3;   sml[23]="     ";
107=   smn[24]="NST";   smt[24]=3;   sml[24]="     ";
108=   smn[25]="CNF";   smt[25]=3;   sml[25]="     ";
109=   smn[26]="---";                  /*killed*/
110=   smn[27]="CDI";   smt[27]=3;   sml[27]="     ";
111=   smn[28]="CDD";   smt[28]=3;   sml[28]="     ";
112=   smn[29]="SUM";   smt[29]=3;   sml[29]="     ";
113=   smn[30]="S01";   smt[30]=4;   sml[30]="     ";
114=   smn[31]="S02";   smt[31]=4;   sml[31]="     ";
115=   smn[32]="S03";   smt[32]=4;   sml[32]="     ";
116=   smn[33]="S04";   smt[33]=4;   sml[33]="     ";
117=   smn[34]="S05";   smt[34]=4;   sml[34]="     ";
118=   smn[35]="S06";   smt[35]=4;   sml[35]="     ";
119=   smn[36]="S07";   smt[36]=4;   sml[36]="     ";
120=   smn[37]="C01";   smt[37]=4;   sml[37]="     ";
```

```
121=   smn[38]="C02";   smt[38]=4;   sml[38]="     ";
122=   smn[39]="C03";   smt[39]=4;   sml[39]="     ";
123=   smn[40]="C04";   smt[40]=4;   sml[40]="     ";
124=   smn[41]="C05";   smt[41]=4;   sml[41]="     ";
125=   smn[42]="C06";   smt[42]=4;   sml[42]="     ";
126=   smn[43]="INF";   smt[43]=3;   sml[43]="     ";
127=   smn[44]="IFS";   smt[44]=3;   sml[44]="     ";
128=   smn[45]="DRI";   smt[45]=1;   sml[45]="     ";
129=   smn[46]="S08";   smt[46]=4;   sml[46]="     ";
130=   smn[47]="$$$";
131=
132=   for(i=0;i < NSTM ;i++)
133=     { for(j=0;j<=4;j++) *(sml[i]+j) = *(sml[0]+j);
134=       for(j=0;j<=9;j++) smd[i][j]=0.0; }
135=}
136=
137=
138=
139=
140=/********** rule-based inference **********/
141=rbinfer(rsno)
142=    char rsno[];
143={
144=   FILE *fprb;
145=   extern char  *smn[];
146=   extern int   smt[];
147=   extern float smd[][10];
148=   extern char  *sml[];
149=   extern int   pp;
150=   char line[100];
151=   char buf3[3], buf4[4];
152=   int  hty, preh, nact, im;
153=   int  i, jj, k,  itype , skip;
154=   float dd, dsatis, dfif, dsif, ssatis, sact=0;
155=   float premise(), min(), max();
156=   float bp[6];       /*basic pro. of no,slig,mode,seve,dest,uk*/
157=
158=   if(pp=='t') printf("      Rule Set # = %s \n",rsno);
159=
160=   fprb=fopen("rbase","r");
161=
162=   do                          /*loop until break*/
163=   {
164=                               /*find associated rule in rule-base*/
165=      k=0;
166=      while (k == 0)
167=         k=findr(fprb,rsno);    /*k=1 if rlue is found*/
168=      if(k <= -1) break;       /*k=-9 if Rule$$.. is detected*/
169=
170=      for (i=0; i<=9; i++) bp[i]=0;
171=      nact=0;   preh=1;   dfif=1;   skip=0;
172=
173=      do                          /*loop until break*/
174=       {
175=         lget(fprb,line);         /* 1 line get*/
176=
177=         hty=header(line,preh); /*header classify*/
178=         preh=hty;
179=         if(hty <= 0) break;     /*undefined header*/
180=
```

```
181=       if(hty==3) dfif=dsatis;          /*THEN IF  set first IF*/
182=       if(hty==3 && dfif<0.1) skip=1;   /*THEN IF &&small first IF*/
183=
184=       if((skip==1 && hty != 2 && hty!=10) || ssatis>0.99)
185=           { dsatis=0;                  /*skip examination*/
186=             goto skipp; }
187=
188=       if(pp=='y' || pp=='t') printf("     %s", line);
189=
190=       if (hty==1 || hty==2)
191=           {dsatis=1; ssatis=0;     /*reset for IF or OR IF*/
192=            skip=0; }               /*no OR IF in present rule base*/
193=
194=       if (hty <= 8)           /*premise clouse*/
195=          dd=premise(line);    /*deg. of satisfaction*/
196=
197=       if(hty==1 || hty==2 || hty==5)     /*IF, IF-and*/
198=               dsatis=min(dsatis,dd);
199=       else if(hty==6)                    /*IF-or*/
200=               dsatis=max(dsatis,dd);
201=
202=       else if(hty==3 || hty==4)            /*THEN IF or ELSE IF*/
203=               {dsif=dd;           /*deg. of second IF clouses*/
204=                dd=min(dfif,dsif);
205=                dsatis=min(dd,(1-ssatis)); }
206=       else if(hty==7)                  /*ELSE IF-and*/
207=               {dsif=min(dsif,dd);
208=                dd=min(dfif,dsif);
209=                dsatis=min(dd,(1-ssatis)); }
210=       else if(hty==8)                  /*ELSE IF-or*/
211=               {dsif=max(dsif,dd);
212=                dd=min(dfif,dsif);
213=                dsatis=min(dd,(1-ssatis)); }
214=  if(pp=='t' && hty<=8)
215=  printf("func.rbinfer:IF:  hty=%d dd=%5.2f ssatis=%5.2f dsatis=%5.2f\n"
216=    ,hty,dd,ssatis,dsatis);    /***test***/
217=
218=       skipp:    ;
219=
220=       if(hty==9 && nact==0)     /*first action - identyfy stm*/
221=          { nact++;
222=            stmove(line,8,3,buf3);
223=            im=findstm(buf3);
224=            itype=smt[im];  }
225=
226=       if(hty==9 && (itype==1 || itype==3))          /*THEN*/
227=            {
228=             action(line,dsatis,&sact,bp);     /*action*/
229=
230=             ssatis=ssatis+dsatis;
231=  if(pp=='t')
232=  printf("func.rbinfer:THEN:  dsatis=%5.2f ssatis=%5.2f sact=%5.2f \n"
233=    ,dsatis, ssatis, sact); /***test***/
234=             nact++;
235=            }
236=
237=       else if(hty==11 && itype==1)             /*THEN-and*/
238=            {action(line,dsatis,&sact,bp);
239=             nact++; }
240=
```

```
241=        else if(hty==9 && itype==2)        /*substitute linguistic data*/
242=             { if(dsatis > 0.99)
243=                 { stmove(line,12,3,buf3);
244=                   if(comp(buf3,"=  ",3)==1)
245=                   stmove(line,17,4,sml[im]);
246=                   ssatis=ssatis+dsatis,  nact++;
247=                   sact=sact+dsatis;
248=       if(pp=='t') psml(); /***test***/
249=              }   }
250=        else if(hty==10)                   /*ELSE*/
251=              bp[5]=1-sact;                   /*unknown*/
252=
253=       }
254=      while(hty > 0);
255=
256=   /*end of one rule.  update short term memory*/
257=
258=   if(pp=='t' && dsatis>0.1)
259=   printf("func.rbinfer: ssatis=%f sact=%f\n", ssatis,sact); /*test***/
260=
261=      if (itype==1 && ssatis>=0.1)        /*neglect small contribution*/
262=            update(im,bp);          /*Dempster's rule of combination*/
263=
264=       else if(itype==3 && ssatis>0.1)            /*for numerical data*/
265=         { smd[im][0]=smd[im][0]+bp[0];
266=           sml[im]="answ"; }
267=
268=      if((pp=='y' || pp=='t') && ssatis>0.1)
269=        { printf("     %s : ", smn[im]);
270=          if(itype==1)
271=             { for(i=0;i<=5;i++) printf("%5.2f     ",bp[i]);
272=               printf("\n           ");
273=               for(i=0;i<=9;i++) printf("%5.2f", smd[im][i]); }
274=          if(itype==3)
275=             { printf("%8.3f\n",bp[0]);
276=               printf("           %8.3f", smd[im][0]); }
277=          if(itype==2)  printf("%s",sml[im]);
278=          printf("\n\n"); }
279=
280=   }
281=   while( k >= 0);
282=
283=   fclose(fprb);
284=}
285=
286=
287=
288=
289=/******** check premise ******** --- return deg. of satisfaction */
290=float premise(line)
291=   char line[];
292={
293=   extern int  smt[];
294=   extern char *sml[];
295=   extern int  pp;
296=   char buf3[3];
297=   int im, itype, unans;
298=   float degs, deg1(), deg2(), deg3(), deg4();
299=
300=   stmove(line,8,3,buf3);
```

```
301=  im=findstm(buf3);     /*find stm no.*/
302=  itype=smt[im];        /*read stm type (1,2,3,4)*/
303=
304=  unans=ccomp( sml[im], "unan", 4);
305=                        /* 1 if unanswered */
306=  if(pp=='t') printf("func.premise:sml[%d]=%s$ unans=%d\n",im,sml[im],un
307=  if(unans==1)  ask(im,itype);  /*issue a question*/
308=
309=  if(itype==1)      degs=deg1(im,line);
310=  else if(itype==2) degs=deg2(im,line);
311=  else if(itype==3) degs=deg3(im,line);
312=  else if(itype==4) degs=deg4(im,line);
313=
314=  if(pp=='t')
315=  printf("func.premise: im=%d  itype=%d  degs=%7.3f \n", im, itype, degs
316=  return(degs);
317=}
318=
319=
320=
321=/********** degree of satisfaction for type-2 **********/
322=float deg2(im,line)
323=  int im;
324=  char line[];
325={
326=  extern char *sml[];
327=  char buf4[4];
328=  int i, j, jj;
329=  float degs;
330=
331=  stmove(line,17,4,buf4);
332=  for(i=0;i<=3;i++)
333=    { if(buf4[i]=='\n' || buf4[i]=='0')
334=        for(j=i;j<=3;j++) buf4[j]=' '; }
335=  /*linguistic data*/
336=  if((jj=comp(buf4, sml[im], 4)) == 1) degs=1;
337=  else degs=0;
338=  return(degs);
339=}
340=
341=
342=/********** degree of satisfaction for type-3 **********/
343=float deg3(im,line)
344=  int  im;
345=  char line[];
346={
347=  extern int   smt[];
348=  extern float smd[][10];
349=  extern char  *sml[];
350=  char buf6[6];
351=  int jj, imm, itype;
352=  float xdata, x, degs=0, convert();
353=
354=  stmove(line,17,6,buf6);
355=
356=  if ((buf6[0] >= '0' && buf6[0] <= '9') || buf6[0] == '-' )
357=                                        /*nunurical data*/
358=       x=convert(buf6);                 /*convert char to float*/
359=
360=  else                          /*character -- search short term memory*/
```

```
361=     { imm=findstm(buf6);
362=       itype=smt[imm];
363=       if (ccomp(sml[imm], "unan",4) == 1)
364=          ask(imm,itype);
365=       x=smd[imm][0]; }
366=
367=  stmove(line,12,2,buf6);
368=  xdata=smd[im][0]; /*smd[im][0] is input numerical data*/
369=
370=  if (ccomp(buf6,">=",2) == 1)
371=    { if (xdata >= x) degs=1; }
372=  else if (ccomp(buf6,"> ",2) == 1)
373=    { if (xdata > x) degs=1; }
374=  else if (ccomp(buf6,"<=",2) == 1)
375=    { if (xdata <= x) degs=1; }
376=  else if (ccomp(buf6,"< ",2) == 1)
377=    { if (xdata < x) degs=1; }
378=
379=  return(degs);              /*degs=0.0 or 1.0*/
380=}
381=
382=
383=/**********  degree of satisfaction for typ-4 **********/
384=float deg4(im,line)
385=  int im;
386=  char line[];
387={
388=  extern float smd[][10];
389=  char buf4[4];
390=  int jj;
391=  float degs;
392=
393=  stmove(line,17,4,buf4);
394=
395=  if ((jj=comp(buf4,"yes",3)) == 1)  degs=smd[im][0];
396=  else if ((jj=comp(buf4,"no",2)) == 1) degs=smd[im][1];
397=  else  degs=0;
398=
399=  return(degs);
400=}
401=
402=
403=/**********  degree of satisfaction for type-1 **********/
404=/*   certaimty measures of no, no-slig, slig,.....  */
405=float deg1(im,line)
406=  int im;
407=  char line[];
408={
409=  extern float smd[][10];
410=  char buf4[4];
411=  int i;
412=  float degs=0, bp[10], flow();
413=
414=  for(i=0; i<=9; i++) bp[i]=smd[im][i];
415=  stmove(line,17,4,buf4);
416=  degs=flow(buf4,bp);          /*lower prob. of fuzzy subset*/
417=
418=  if (line[21] == '+')
419=      {stmove(line,22,4,buf4);
420=       degs=degs+flow(buf4,bp); }
```

```
421=
422=  return(degs);
423=}
424=/*************** lower prob. of fuzzy subset*****************/
425=float flow(buf,bp)
426=  char buf[];
427=  float bp[];
428={
429=  float lp;
430=
431=  if (ccomp(buf,"no",2) == 1)      lp=bp[0]+bp[1];
432=  else if (ccomp(buf,"slig",2) == 1) lp=bp[1]+bp[2]+bp[3];
433=  else if (ccomp(buf,"mode",2) == 1) lp=bp[3]+bp[4]+bp[5];
434=  else if (ccomp(buf,"seve",2) == 1) lp=bp[5]+bp[6]+bp[7];
435=  else if (ccomp(buf,"dest",2) == 1) lp=bp[7]+bp[8];
436=  else  lp=0;
437=
438=  return(lp);
439=}
440=
441=/********** lower prob. ********************/
442=float lp(im,i)
443=  int im, i;
444={
445=  extern float smd[][10];
446=  float llp;
447=  if(i==0) llp=smd[im][0]+smd[im][1];
448=  if(i==1) llp=smd[im][1]+smd[im][2]+smd[im][3];
449=  if(i==2) llp=smd[im][3]+smd[im][4]+smd[im][5];
450=  if(i==3) llp=smd[im][5]+smd[im][6]+smd[im][7];
451=  if(i==4) llp=smd[im][7]+smd[im][8];
452=  if(i==5) llp=smd[im][9];                    /*uk*/
453=  return(llp);
454=}
455=
456=
457=
458=/********** issue a question and get data **********/
459=ask(im,itype)
460=  int im, itype;
461={
462=  extern char  *smn[];
463=  extern float  smd[][10];
464=  extern char  *sml[];
465=  extern int pp;
466=  FILE *fpq;
467=  char line[100], buf4[4];
468=  int i, j, jj, jk;
469=  float dd;
470=
471=  if ((itype<=1) || (itype>=5)) return;
472=
473=  fpq=fopen("quest","r");          /*stored file is "quest"*/
474=
475=  jk=0;
476=  while(jk==0)                     /*search desired question*/
477=   {
478=    lget(fpq,line);
479=    if (ccomp(line,"Q-",2) == 1)
480=       {strmove(line,2,3,buf4);
```

```
481=          if (ccomp(buf4, smn[im], 3) == 1) jk=1;    /*find*/
482=          else if(ccomp(buf4,"$$$",3) == 1) jk = -1;     /*no quest*/
483=         }
484=     }
485=   if (jk == -1)
486=      {printf("no question setence in file-quest\n");
487=       return; }
488=
489=   lget(fpq,line);
490=   do { printf("%s", line);          /*print question*/
491=        lget(fpq,line); }
492=   while (ccomp(line,"Q-",2) != 1);
493=
494=   fclose(fpq);
495=   printf("===");
496=   stmove("an##",0,4,sml[im]);
497=
498= /* * * data input * * */
499=   if(itype==2)                        /*linguistic data*/
500=     {scanf("%s",line);
501=      for(i=0;i<=3;i++)
502=        { if(line[i]=='\n' || line[i]=='\0')
503=            for(j=i;j<=3;j++) line[j]=' ';
504=        }
505=      stmove(line,0,4,sml[im]);
506=     }
507=   else if(itype==3)                   /*numerical data*/
508=     {scanf("%f", &dd);
509=      smd[im][0]=dd;
510=     }
511=   else if(itype==4)                   /*yes, grey, no, uk*/
512=     {scanf("%s",line);
513=      if ((jj=comp(line,"yes",1)) == 1)
514=         smd[im][0]=1;
515=      if ((jj=comp(line,"grey",1)) ==1)
516=         {smd[im][0]=0.6;
517=          smd[im][1]=0.2; }          /*sum is not 1.0*/
518=      if ((jj=comp(line,"no",1)) == 1)
519=          smd[im][1]=1;
520=      if ((jj=comp(line,"unknown",1)) ==1)
521=          smd[im][2]=1;
522=     }
523=}
524=
525=
526=
527=
528=/**********  action THEN clouse **********/
529=action(line, dsatis, sact, bp)
530=   char line[];
531=   float dsatis, *sact, bp[];
532={
533=   char buf4[4], buf6[6];
534=   float cm, convert();
535=
536=   if (dsatis < 0.1) return;           /*neglect small comtribution*/
537=
538=   stmove(line,17,6,buf6);
539=   cm=convert(buf6);                   /*convert written cm to float*/
540=   cm=cm*dsatis;
```

```
541=
542=   if (cm < 0.1) return;                /*neglect small comtribution*/
543=
544=   *sact = *sact + cm;
545=   stmove(line,12,4,buf4);
546=
547=   if      (ccomp(buf4,"no  ",2) == 1)
548=       bp[0]=bp[0]+cm;
549=   else if(ccomp(buf4,"slig",2) == 1)
550=       bp[1]=bp[1]+cm;
551=   else if(ccomp(buf4,"mode",2) == 1)
552=       bp[2]=bp[2]+cm;
553=   else if(ccomp(buf4,"seve",2) == 1)
554=       bp[3]=bp[3]+cm;
555=   else if(ccomp(buf4,"dest",2) == 1)
556=       bp[4]=bp[4]+cm;
557=   else if(ccomp(buf4,"uk  ",2) == 1)
558=       bp[5]=bp[5]+cm;
559=
560=   else if(ccomp(buf4,"=   ",2) == 1)      /*not certainty measure*/
561=       bp[0] = cm ;                  /*substitute numerical to smd[im][0]*/
562=   else if(ccomp(buf4,"++  ",2) == 1)
563=       bp[0] = bp[0] + cm;           /*accumulate*/
564=}
565=
566=
567=
568=/********** update short term memory **********/
569=/*   Dempster's rule of combination for fuzzy subset */
570=update(im, bp)
571=   int im;
572=   float bp[];
573={
574=   extern float smd[][10];
575=   extern char  *sml[];
576=   int i, ii, jj;
577=   float dbp[10], pp[10], sum;
578=
579=   if(bp[5] > 0.95) return;                  /*almost unknown*/
580=
581=   sum=0;
582=   for(i=0;i<=5;i++) sum=sum+bp[i];
583=   if(sum<0.999) bp[5]=bp[5]+(1-sum);        /*increase unknown*/
584=
585=   if(ccomp(sml[im],"unan",4) == 1)  /*first update*/
586=     {for(i=0;i<=5;i++)                          /*simple insert*/
587=        { ii = 2*i;
588=          if(ii == 10) ii=9;
589=          smd[im][ii]=bp[i];
590=        }
591=      stmove("an$$",0,4,sml[im]);
592=     }
593=
594=   else            /*Dempster's rule of combination for fuzzy subset*/
595=     {
596=      for(i=0;i<=9;i++) dbp[i]=smd[im][i];
597=
598=      for(i=0;i<=4;i++)
599=        {ii=2*i;
600=         pp[ii]=dbp[ii]*bp[i]+dbp[9]*bp[i]+dbp[ii]*bp[5];
```

```
601=        }
602=      pp[9]=dbp[9]*bp[5];                          /*unknown set*/
603=
604=      for(i=0;i<=4;i++)                            /*intersect fuzzy set*/
605=        {ii=2*i-1;
606=         pp[ii]=0.5*(dbp[ii-1]*bp[i+1]+dbp[ii+1]*bp[i]);
607=        }
608=
609=      sum=0;                    /*normalize*/
610=      for(i=0;i<=8;i++) sum=sum+pp[i];
611=      if(sum < 0.05)            /*no normalize if small certainty*/
612=          pp[9]=1.0-sum;
613=      else
614=         {sum=sum+pp[9];
615=          for(i=0;i<=9;i++) pp[i]=pp[i]/sum;
616=         }
617=
618=      for(i=0;i<=9;i++)         /*write to short term memory*/
619=         smd[im][i]=pp[i];
620=     }
621=}
622=
623=
624=
625=/********** report the result **********/
626=result()
627={
628=  float  lp(),  lpmax=0;
629=  int  i, j, im;
630=
631=  im=findstm("FIN");
632=
633=  for(i=0;i<=4;i++)              /*search max*/
634=     { if( lp(im,i) > lpmax)
635=        { lpmax=lp(im,i);
636=          j=i; }
637=     }
638=
639=  printf("\n\n************************************************\n\n")
640=
641=  if(lpmax < 0.2)
642=    printf("  An appropriate anser is not obtained.\n");
643=
644=  else
645=    {if(j == 0) printf("  There is no damage.\n");
646=     if(j == 1) printf("  The damage is slight.\n");
647=     if(j == 2) printf("  The damage is moderate.\n");
648=     if(j == 3) printf("  The damage is severe.\n");
649=     if(j == 4) printf("  The damage is destructive.\n");
650=    }
651=
652=  printf("\n************************************************\n\n");
653=
654=  printf("          no   slight moderate   severe destract   unkown \n")
655=  printf("FIN");
656=  for(i=0;i<=5;i++) printf("%9.4f",lp(im,i));
657=  im=findstm("GLO");
658=  printf("\nGLO");
659=  for(i=0;i<=5;i++) printf("%9.4f",lp(im,i));
660=  im=findstm("LOC");
```

```
661=  printf("\nLOC");
662=  for(i=0;i<=5;i++) printf("%9.4f",lp(im,i));
663=
664=  printf("\n\n****************** End of SPERIL ********************\n");
665=}
666=
667=
668=
669=/**********  find rule **********/
670=/*    return 1   :find*/
671=/*           -1  :EOF */
672=/*           0   :otherwise */
673=findr(fp,rsno)
674=  FILE *fp;
675=  char rsno[];      /*rule set no.*/
676={
677=  int jj, k;
678=  char line[100], buf2[2];
679=  extern int pp;
680=
681=  lget(fp,line);    /*one line get*/
682=
683=  if((jj=line[0]) == EOF) k = -1;
684=  else
685=    {
686=     if(ccomp(line,"Rule",4) != 1) k=0;
687=     else
688=       {stmove(line,4,2,buf2);
689=        if(ccomp(buf2,rsno,2) == 1) k=1;
690=        else if(ccomp(buf2,"$$",2) == 1) k = -9;
691=        else k=0;
692=       }
693=    }
694=  if((pp=='y' || pp=='t') && k==1) printf("     %s", line);
695=  return(k);
696=}
697=
698=
699=
700=/**********  header classification **********/
701=/*    return type=1,2,3,4,5,6,7,8,9,10,11,12 */
702=header(line,pr)
703=  char line[];
704=  int  pr;      /*previous header type*/
705={
706=  int  jj, k = -9;
707=
708=  if      (ccomp(line,"      IF",7) == 1) k=1;
709=  else if(ccomp(line,"  OR IF",7) == 1) k=2;
710=  else if(ccomp(line,"THEN IF",7) == 1) k=3;
711=  else if(ccomp(line,"ELSE IF",7) == 1) k=4;
712=
713=  else if(ccomp(line,"    and",7) == 1)
714=    {if(pr==1 || pr==2 || pr==5 || pr==6) k=5;    /*IF--ann*/
715=     else if(pr==3 || pr==4 || pr==7 || pr==8) k=7;
716=                                                  /*ELSE IF--and*/
717=     else if(pr==9 || pr==10) k=11;               /*THEN--and*/
718=    }
719=  else if(ccomp(line,"     or",7) == 1)
720=    {if(pr==1 || pr==2 || pr==5 || pr==6) k=6;   /*IF--or*/
```

```
721=      else if(pr==3 || pr==4 || pr==7 || pr==8) k=8;
722=                                                  /*ELSE IF--or*/
723=     }
724=
725=   else if(ccomp(line,"    THEN",7) == 1) k=9;
726=   else if(ccomp(line,"    ELSE",7) == 1) k=10;
727=
728=   else if(comp(line,"        ",7) != 1)
729=      { k = -10;
730=        printf("** undefined header -- miss writing of rule base **\n");
731=
732=   return(k);
733=}
734=
735=
736=
737=/********** get one line from file **********/
738=lget(fp,line)
739=   FILE *fp;
740=   char line[];
741={
742=   int i=0, j;
743=   int c='a';
744=
745=   while (c != '\n')
746=     { c=getc(fp);          /*getc is in stdio.h */
747=       line[i]=c;
748=       i++; }
749=   line[i] = '\0';
750=
751=   i++;
752=   if(i < 50)
753=     { for(j=0; j<=10; j++) line[i+j]=' '; }    /*insert 11 blank at tail*/
754=}
755=
756=
757=
758=/********** string compare **********/
759=/*    return  1 if coincide */
760=comp(s,ss,n)
761=   char s[], ss[];
762=   int n;
763={
764=   int i, k=1;
765=
766=   for (i=0; i<n; i++)
767=     { if(s[i]!=ss[i] && s[i]!=' ' && ss[i]!=' ')
768=          { if(s[i]!='\n' || s[i]!='\0')
769=               {k=0;  break; }  }
770=     }
771=   return(k);
772=}
773=
774=ccomp(s,ss,n)
775=    char s[], ss[];
776={
777=    int i, k=1;
778=
779=    for(i=0; i<n; i++)
780=       if(s[i] != ss[i]) {k=0; break; }
```

```
781=    return(k);
782=}
783=
784=
785=/********** string window move **********/
786=stmove(s,ist,n,ss)
787=  char s[], ss[];
788=  int ist, n;
789={
790=  int i;
791=  for (i=0; i<n; i++)  ss[i]=s[ist+i];
792=}
793=
794=
795=
796=/********** find short term memory **********/
797=/*    return  im (stm[im])      */
798=findstm(na)
799=  char na[];
800={
801=  extern *smn[];
802=  int im, j=0, jj=0;
803=
804=  while(jj != 1)
805=    { if((jj=comp(smn[j],na,3)) == 1)  im=j;
806=      else if((jj=comp(smn[j],"$$$",3)) ==1)  im = -1;
807=      j++;
808=    }
809=  if(im == -1)
810=     printf("  %s  is not in short term memory.  -- error\n", na);
811=  return(im);
812=}
813=
814=
815=
816=/********** convert char[] to float **********/
817=float convert(buf)
818=  char buf[];
819={
820=  float x=0.0, y=0.0;
821=  int   i, j, ii=0, sign=1;
822=
823=  if (buf[0] == '-') { sign = -1;  ii=1; }
824=
825=  for(i=ii; buf[i]>='0' && buf[i]<='9'; i++)
826=      x=10.0*x+(buf[i]-'0');
827=
828=  if(buf[i] == '.')
829=    { for(j=i+1; buf[j]>='0' && buf[j]<='9'; j++)
830=         y=y+(buf[j]-'0')/(10.0*(j-i));
831=    }
832=
833=  x=(x+y)*sign;
834=  return(x);
835=}
836=
837=
838=
839=/********** min **********/
840=float min(x,y)
```

```
841=  float x, y;
842={
843=  if(x>y) return(y);
844=  else return(x);
845=}
846=
847=/*********** max ***********/
848=float max(x,y)
849=   float x, y;
850={
851=   if(x>y) return(x);
852=   else return(y);
853=}
854=
855=/*********** ptintf sml[im] fo test *****************/
856=psml()
857={
858=  extern *sml[];
859=  int i;
860=  for(i=0;i<NSTM;i++)
861=    printf(" sml[%d]=%s",i,sml[i]);
862=  printf("\n");
863=}
```

APPENDIX B RULE-BASE: rbase

```
 1=Rule0101
 2=      IF: GLO is   dest
 3=    THEN: FIN dest 1
 4=ELSE IF: GLO is   seve
 5=    THEN: FIN seve 1
 6=ELSE IF: GLO is   mode+slig
 7=      or: LOC is   dest+seve
 8=    THEN: FIN mode 1
 9=ELSE IF: LOC is   mode+slig
10=    THEN: FIN slig 1
11=ELSE IF: GLO is   no
12=     and: LOC is   no
13=    THEN: FIN no   1
14=    ELSE: FIN uk
15=
16=Rule0201
17=      IF: MAT is   r/c
18=THEN IF: STI is   dest
19=    THEN: GLO dest 0.6
20=ELSE IF: STI is   seve
21=    THEN: GLO seve 0.6
22=ELSE IF: STI is   mode
23=    THEN: GLO mode 0.6
24=ELSE IF: STI is   slig
25=    THEN: GLO slig 0.6
26=ELSE IF: STI is   no
27=    THEN: GLO no   0.6
28=    ELSE: GLO uk
29=
30=Rule0202
31=      IF: MAT is   steel
32=THEN IF: STI is   dest
33=    THEN: GLO dest 0.8
34=ELSE IF: STI is   seve
35=    THEN: GLO seve 0.8
36=ELSE IF: STI is   mode
37=    THEN: GLO mode 0.8
38=ELSE IF: STI is   slig
39=    THEN: GLO slig 0.8
40=ELSE IF: STI is   no
41=    THEN: GLO no   0.8
42=    ELSE: GLO uk
43=
44=Rule0203
45=      IF: DAM is   dest
46=    THEN: GLO dest 0.4
47=ELSE IF: DAM is   seve
48=    THEN: GLO seve 0.4
49=ELSE IF: DAM is   mode
50=    THEN: GLO mode 0.4
51=ELSE IF: DAM is   slig
52=    THEN: GLO slig 0.4
53=ELSE IF: DAM is   no
54=    THEN: GLO no   0.4
55=    ELSE: GLO uk
56=
57=Rule0204
58=      IF: DRI is   dest
59=    THEN: GLO dest 0.6
60=ELSE IF: DRI is   seve
```

```
 61=    THEN: GLO seve 0.6
 62=ELSE IF: DRI is   mode
 63=    THEN: GLO mode 0.6
 64=ELSE IF: DRI is   slig
 65=    THEN: GLO slig 0.6
 66=ELSE IF: DRI is   no
 67=    THEN: GLO no   0.6
 68=    ELSE: GLO uk
 69=
 70=Rule0205
 71=      IF: INS is   careful
 72=THEN IF: VST is   dest
 73=    THEN: GLO dest 1
 74=ELSE IF: VST is   seve
 75=    THEN: GLO seve 1
 76=ELSE IF: VST is   mode
 77=    THEN: GLO mode 0.9
 78=ELSE IF: VST is   slig
 79=    THEN: GLO slig 0.8
 80=ELSE IF: VST is   no
 81=    THEN: GLO no   0.8
 82=    ELSE: GLO uk
 83=
 84=Rule0206
 85=      IF: INS is   ml=more-or-less
 86=THEN IF: VST is   dest
 87=    THEN: GLO dest 1
 88=ELSE IF: VST is   seve
 89=    THEN: GLO seve 1
 90=ELSE IF: VST is   mode
 91=    THEN: GLO mode 0.9
 92=ELSE IF: VST is   slig
 93=    THEN: GLO slig 0.7
 94=ELSE IF: VST is   no
 95=    THEN: GLO no   0.6
 96=    ELSE: GLO uk
 97=
 98=Rule0207
 99=      IF: INS is   rough
100=      or: INS is   uk
101=THEN IF: VST is   dest
102=    THEN: GLO dest 1
103=ELSE IF: VST is   seve
104=    THEN: GLO seve 1
105=ELSE IF: VST is   mode
106=    THEN: GLO mode 0.8
107=ELSE IF: VST is   slig
108=    THEN: GLO slig 0.6
109=ELSE IF: VST is   no
110=    THEN: GLO no   0.4
111=    ELSE: GLO uk
112=
113=Rule0401
114=      IF: VNS is   dest
115=    THEN: LOC dest 0.7
116=ELSE IF: VNS is   seve
117=    THEN: LOC seve 0.7
118=ESLE IF: VNS is   mode
119=    THEN: LOC mode 0.7
120=ELSE IF: VNS is   slig
```

```
121=    THEN:LOC slig 0.7
122=ELSE IF:VNS is    no
123=    THEN:LOC no   0.7
124=    ELSE:LOC uk
125=
126=Rule0402
127=      IF:DRI is   dest
128=    THEN:LOC dest 0.7
129=ELSE IF:DRI is    seve
130=    THEN:LOC seve 0.7
131=ELSE IF:DRI is    mode
132=    THEN:LOC mode 0.7
133=ELSE IF:DRI is    slig
134=    THEN:LOC slig 0.7
135=ELSE IF:DRI is    no
136=    THEN:LOC no   0.7
137=    ELSE:LOC uk
138=
139=Rule0501
140=      IF:MAT is   r/c
141=THEN IF:ISD <=    -8.9
142=    THEN:DRI uk   1
143=ELSE IF:ISD <=    0.4
144=    THEN:DRI no   0.9
145=ELSE IF:ISD <=    0.8
146=    THEN:DRI slig 0.9
147=ELSE IF:ISD <=    1.3
148=    THEN:DRI mode 0.9
149=ELSE IF:ISD <=    2.0
150=    THEN:DRI seve 0.9
151=ELSE IF:ISD >     2.0
152=    THEN:DRI dest 0.9
153=    ELSE:DRI uk
154=
155=Rule0502
156=      IF:MAT is   r/c
157=    THEN:FLX =    no
158=
159=Rule0503
160=      IF:MAT is   steel
161=     and:NST >=   8
162=     and:FLX is   yes
163=THEN IF:ISD <=    0.6
164=    THEN:DRI no   0.8
165=ELSE IF:ISD <=    1.2
166=    THEN:DRI slig 0.8
167=ELSE IF:ISD <=    1.9
168=    THEN:DRI mode 0.8
169=ELSE IF:ISD <=    3.0
170=    THEN:DRI seve 0.8
171=ELSE IF:ISD >     3.0
172=    THEN:DRI dest 0.8
173=    ELSE:DRI uk
174=
175=Rule0504
176=      IF:NST <    8
177=      or:FLX is   no  +uk
178=     and:MAT is   steel
179=THEN IF:ISD <=    0.4
180=    THEN:DRI no   0.8
```

```
181=ELSE IF: ISD <=   0.9
182=    THEN: DRI slig 0.8
183=ELSE IF: ISD <=   1.5
184=    THEN: DRI mode 0.8
185=ELSE IF: ISD <=   2.3
186=    THEN: DRI seve 0.8
187=ELSE IF: ISD >    2.3
188=    THEN: DRI dest 0.8
189=    ELSE: DRI uk
190=
191=Rule0601
192=      IF: MAT is    r/c
193=THEN IF: NST <=    4
194=    THEN: IFS =     2.0   (8.0/NST)
195=ELSE IF: NST <=    6
196=    THEN: IFS =     1.33
197=ELSE IF: NST <=    8
198=    THEN: IFS =     1.0
199=ELSE IF: NST <=    11
200=    THEN: IFS =     0.72
201=ELSE IF: NST <=    14
202=    THEN: IFS =     0.57
203=ELSE IF: NST <=    18
204=    THEN: IFS =     0.44
205=ELSE IF: NST <=    23
206=    THEN: IFS =     0.32
207=ELSE IF: NST <=    30
208=    THEN: IFS =     0.27
209=ELSE IF: NST <=    40
210=    THEN: IFS =     0.2
211=ELSE IF: NST >     40
212=    THEN: IFS =     0.17
213=
214=Rule0602
215=      IF: MAT is    steel
216=THEN IF: NST <=    4
217=    THEN: IFS =     1.62   (6.5/NST)
218=ELSE IF: NST <=    6
219=    THEN: IFS =     1.1
220=ELSE IF: NST <=    8
221=    THEN: IFS =     0.81
222=ELSE IF: NST <=    11
223=    THEN: IFS =     0.59
224=ELSE IF: NST <=    14
225=    THEN: IFS =     0.46
226=ELSE IF: NST <=    18
227=    THEN: IFS =     0.36
228=ELSE IF: NST <=    23
229=    THEN: IFS =     0.28
230=ELSE IF: NST <=    30
231=    THEN: IFS =     0.22
232=ELSE IF: NST <=    40
233=    THEN: IFS =     0.163
234=ELSE IF: NST <=    55
235=    THEN: IFS =     0.118
236=ESLE IF: NST <=    70
237=    THEN: IFS =     0.093
238=ELSE IF: NST >     70
239=    THEN: IFS =     0.075
240=
```

```
241=Rule0603
242=       IF:MAT is    r/c
243=      and:INF >=    IFS
244=THEN  IF:CNF <     -8.9
245=     THEN:STI uk    1
246=ELSE  IF:CNF <=    4
247=     THEN:STI no    0.9
248=ELSE  IF:CNF <=    12
249=     THEN:STI slig  0.9
250=ELSE  IF:CNF <=    22
251=     THEN:STI mode  0.9
252=ELSE  IF:CNF <=    40
253=     THEN:STI seve  0.9
254=ELSE  IF:CNF >     40
255=     THEN:STI dest  0.9
256=     ELSE:STI uk
257=
258=Rule0604
259=       IF:MAT is    r/c
260=      and:INF <     IFS
261=THEN  IF:CNF <     -8.9
262=     THEN:STI uk    1
263=ELSE  IF:CNF <=    6
264=     THEN:STI slig  0.9
265=ELSE  IF:CNF <=    15
266=     THEN:STI mode  0.9
267=ELSE  IF:CNF <=    30
268=     THEN:STI seve  0.9
269=ELSE  IF:CNF >     30
270=     THEN:STI dest  0.9
271=     ELSE:STI uk
272=
273=Rule0605
274=       IF:MAT is    steel
275=      and:INF >=    IFS
276=THEN  IF:CNF <     -8.9
277=     THEN:STI uk    1
278=ELSE  IF:CNF <=    4
279=     THEN:STI no    0.9
280=ELSE  IF:CNF <=    12
281=     THEN:STI slig  0.9
282=ELSE  IF:CNF <=    22
283=     THEN:STI mode  0.9
284=ELSE  IF:CNF <=    40
285=     THEN:STI seve  0.9
286=ELSE  IF:CNF >     40
287=     THEN:STI dest  0.9
288=     ELSE:STI uk
289=
290=Rule0606
291=       IF:MAT is    steel
292=      and:INF <     IFS
293=THEN  IF:CNF <     -8.9
294=     THEN:STI uk    1
295=ELSE  IF:CNF <=    6
296=     THEN:STI slig  0.9
297=ELSE  IF:CNF <=    15
298=     THEN:STI mode  0.9
299=ELSE  IF:CNF <=    30
300=     THEN:STI seve  0.9
```

```
301=ELSE IF: CNF >    30
302=    THEN: STI dest 0.9
303=    ELSE: STI uk
304=
305=Rule0701
306=      IF: CDI <    -8.9
307=      or: CDD <    -8.9
308=    THEN: DAM uk   1
309=ELSE IF: CDD <    3   (no decrease)
310=     and: CDI <=   10
311=    THEN: DAM no   0.8
312=ELSE IF: CDD <    3
313=     and: CDI <=   40
314=    THEN: DAM slig 0.8
315=ELSE IF: CDD <=   15
316=    THEN: DAM mode 0.8
317=ELSE IF: CDD <=   40
318=    THEN: DAM seve 0.8
319=ELSE IF: CDD >    40
320=    THEN: DAM dest 0.8
321=    ELSE: DAM uk
322=
323=Rule0901
324=      IF: MAT is   steel
325=THEN IF: S01 is    yes     (partial collaps)
326=    THEN: VST dest 1
327=ELSE IF: S02 is    yes     (buckling of column)
328=    THEN: VST dest 0.5
329=     and: VST seve 0.5
330=ELSE IF: S03 is    yes     (buckling of girder/beam)
331=      or: S04 is    yes     (buckling of diagnal bracing)
332=      or: S05 is    yes     (deformation or loosing of joint)
333=    THEN: VST seve 0.9
334=ELSE IF: S06 is    yes    (spalling/crack on shear wall)
335=    THEN: VST mode 0.8
336=ELSE IF: S07 is    yes    (spalling/crack on exteria/interia wall)
337=      or: S08 is    yes    (spalling/crack on floor)
338=    THEN: VST mode 0.5
339=     and: VST slig 0.5
340=ELSE IF: S01 is    no
341=     and: S02 is    no
342=     and: S03 is    no
343=     and: S04 is    no
344=     and: S05 is    no
345=     and: S06 is    no
346=     and: S07 is    no
347=     and: S08 is    no
348=    THEN: VST no   1
349=    ELSE: VST uk
350=
351=Rule0902
352=      IF: MAT is   r/c
353=THEN IF: C01 is    yes    (partial collaps)
354=    THEN: VST dest 1
355=ELSE IF: C02 is    yes    (large spalling on column)
356=    THEN: VST dest 0.5
357=     and: VST seve 0.5
358=ELSE IF: C03 is    yes    (large spa. on load-bear./shear wall or girder/b
359=    THEN: VST seve 0.8
360=ELSE IF: C04 is    yes    (small spa. on load-bear./shear wall or girder/b
```

```
361=    THEN:VST seve 0.5
362=     and:VST mode 0.5
363=ELSE IF:C05 is    yes     (small cracks on load-bear./shear wall or girder
364=      or:C06 is    yes     (spalling or large clacks on oter walls or floor
365=    THEN:VST mode 0.5
366=     and:VST slig 0.5
367=ELSE IF:C01 is    no
368=     and:C02 is    no
369=     and:C03 is    no
370=     and:C04 is    no
371=     and:C05 is    no
372=     and:C06 is    no
373=    THEN:VST no    1
374=    ELSE:VST uk
375=
376=Rule1001
377=       IF:N01 is    severe     (nonstractural partition)
378=    THEN:SUM ++     10
379=ELSE IF:N01 is    considerable
380=    THEN:SUM ++     5
381=ELSE IF:N01 is    slight
382=    THEN:SUM ++     2.5
383=
384=Rule1002
385=       IF:N02 is    severe    (windows)
386=    THEN:SUM ++     10
387=ELSE IF:N02 is    considerable
388=    THEN:SUM ++     5
389=ELSE IF:N02 is    slight
390=    THEN:SUM ++     2.5
391=
392=Rule1003
393=       IF:N03 is    severe    (doors)
394=    THEN:SUM ++     10
395=ELSE IF:N03 is    considerable
396=    THEN:SUM ++     5
397=ELSE IF:N03 is    slight
398=    THEN:SUM ++     2.5
399=
400=Rule1004
401=       IF:N04 is    severe    (ceiling and light fixtures)
402=    THEN:SUM ++     10
403=ELSE IF:N04 is    considerable
404=    THEN:SUM ++     5
405=ELSE IF:N04 is    slight
406=    THEN:SUM ++     2.5
407=
408=Rule1005
409=       IF:N05 is    severe    (stairs)
410=    THEN:SUM ++     10
411=ELSE IF:N05 is    considerable
412=    THEN:SUM ++     5
413=ELSE IF:N05 is    slight
414=    THEN:SUM ++     2.5
415=
416=Rule1006
417=       IF:N06 is    severe    (elevator)
418=    THEN:SUM ++     10
419=ELSE IF:N06 is    considerable
420=    THEN:SUM ++     5
```

```
421=ELSE IF:N06 is    slight
422=    THEN:SUM ++   2.5
423=
424=Rule1007
425=      IF:N07 is   severe    (electricity)
426=    THEN:SUM ++   10
427=ELSE IF:N07 is    considerable
428=    THEN:SUM ++   5
429=ELSE IF:N07 is    slight
430=    THEN:SUM ++   2.5
431=
432=Rule1008
433=      IF:N08 is   severe    (air coditioning)
434=    THEN:SUM ++   10
435=ELSE IF:N08 is    considerable
436=    THEN:SUM ++   5
437=ELSE IF:N08 is    slight
438=    THEN:SUM ++   2.5
439=
440=Rule1009
441=      IF:N09 is   severe    (water works)
442=    THEN:SUM ++   10
443=ELSE IF:N09 is    considerable
444=    THEN:SUM ++   5
445=ELSE IF:N09 is    slight
446=    THEN:SUM ++   2.5
447=
448=Rule1010
449=      IF:N10 is   severe    (gas facility)
450=    THEN:SUM ++   10
451=ELSE IF:N10 is    considerable
452=    THEN:SUM ++   5
453=ELSE IF:N10 is    slight
454=    THEN:SUM ++   2.5
455=
456=Rule1011
457=      IF:N11 is   severe    (communication)
458=    THEN:SUM ++   10
459=ELSE IF:N11 is    considerable
460=    THEN:SUM ++   5
461=ELSE IF:N11 is    slight
462=    THEN:SUM ++   2.5
463=
464=Rule1012
465=      IF:SUM <=   3
466=    THEN:VNS no   0.8
467=ELSE IF:SUM <=    15
468=    THEN:VNS slig 0.8
469=ELSE IF:SUM <=    35
470=    THEN:VNS mode 0.8
471=ELSE IF:SUM <=    60
472=    THEN:VNS seve 0.9
473=ELSE IF:SUM >     60
474=    THEN:VNS dest 1
475=    ELSE:VNS uk
476=
477=Rule$$$$      mark for end of rules ****************
478=
479=
480= dest : destructive
```

```
481= seve : severe
482= mode : moderate
483= slig : slight
484= no   : no
485= uk   : unknown
486= r/c  : reinforced concrete
487=
488= FIN  : final goal -- damage state
489= GLO  : damage of global nature
490= CUM  : damage of cumulative nature
491= LOC  : damage of local nature
492= DRI  : damage due to drifting
493= STI  : damage of stiffness
494= DAM  : damage of damping
495= FAT  : fatigue & creep
496= VST  : visual damage of stractural member
497= VNT  : visual damage of nonstractural menber
498= MAT  : material of stracture
499= ISD  : interstory drift
500= CNF  : change of natural frequency of vibration
501= INF  : initial natural frequency
502= IFS  : reference of natural frequency
503= CDI  : change of damping -- increase
504= CDD  : change of damping -- decrease
505= FLX  : flexible design
506= INS  : careful visual inspection
507= SUM  : buffer memory for calculation of VNS
508=
509= S01  : check items of visual stractural damage for steel
510=  |
511= S07
512= C01  : check items of visual stractural damage for R/C
513=  |
514= C06
515= N01  : check items of visual nonstractural damage
516=  |
517= N11
```

APPENDIX C QUESTION LIST : quest

```
1=Q-MAT
2=What is the material of the building { r/c(reinforced concrete), steel }?
3=Q-INS
4=Was the visual inspection done craefully? Did you check even inside
5=the covers?  { care(careful), ml(more or less), roug(rough), uk(unknown)
6=Q-FLX
7=Is the building designed to be flexible? { y(yes), n(no), uk(unknown) }
8=Q-ISD
9=What was the maximum interstory drift during the earthquake?
10={ drift/height*100 [%] ,  -9(unknown) }
11=Q-NST
12=Enter the number of stories.
13=Q-INF
14=What was the initial natural frequency [Hz] of the vibration? { -9(unknou
15=Q-CNF
16=What was the decrease [%] of the natural frequency during the eathquake?
17= { -9(unknown) }
18=Q-CDI
19=What was the increase [%] of damping? { 0(almost no increase), -9(unknowr
20=Q-CDD
21=What was the decrease [%] of damping from its peak value?
22=If no initial increase process was observed, use initial value instead
23=of the peak value.  { 0(almost no increase), -9(unknown) }
24=Q-S01
25=Is there partial collaps observed? { y(yes), g(grey), n(no), uk(unknown)
26=Q-S02
27=Is there the buckling of column? { y, g, n, uk }
28=Q-S03
29=Is there the buckling of girder/beam? { y, g, n, uk }
30=Q-S04
31=Is there the buckling of diagnal bracing? { y, g, n, uk }
32=Q-S05
33=Is there the deformation or loosing of joint? { y, g, n, uk }
34=Q-S06
35=Are there considerable spallings/cracks observed on shear wall?
36= { y(yes), g(grey), n(no), uk(unknown) }
37=Q-S07
38=Are there considerable spallings/cracks observed on other exteria/
39=inreria walls? { y, g, n, uk }
40=Q-S08
41=Are there considerable spallings/cracks obsered on floors? { y, g, n, uk
42=Q-C01
43=Is there partial collaps observed? { y(yes), g(grey), n(no), uk(unknown)
44=Q-C02
45=Is there large spalling observed on column? { y, g, n, uk }
46=Q-C03
47=Is there large spalling observed on load bearing wall, shear wall, or
48=girder/beam? { y, g, n, uk }
49=Q-C04
50=Are there small spallings or large cracks observed on column,
51=load bearing wall or girder/beam? { y, g, n, uk }
52=Q-C05
53=Are there considerable number of small cracks observed on column, load
54=bearing wall or girder/beam? { y, g, n, uk }
55=Q-C06
56=Are there spallings or large cracks observed on other walls or floors?
57=  { y, g, n, uk }
58=Q-N01
59=How is the damage of nonstructural partitions?
60= { se(severe), co(considerable) sl(slight), no , uk(unknown) }
```

```
61= {    severe       : more than 10 %    }
62= {    considerable: approximately 5 % }
63=Q-N02
64=How is the damage of windows?  { se, co, sl, no, uk }
65=Q-N03
66=How is the damage of doors?  { se, co, sl, no, uk }
67=Q-N04
68=How is the damage of ceilings including light fixtures?
69=  { se, co, sl, uk }
70=Q-N05
71=How is the damage of stairs? { se, co, sl, no, uk }
72=Q-N06
73=How is the damage of elevators?
74=  { se(severe), co(considerable), sl(slight), no , uk(unknown) }
75=Q-N07
76=How is the damage of electricity?  { se, co, sl, no, uk }
77=Q-N08
78=How is the damage of air conditioning?  { se, co, sl, no, uk }
79=Q-N09
80=How is the damage of waterworks?  { se, co, sl, no, uk }
81=Q-N10
82=How is the damage of gas facility?  { se, co, sl, no, uk }
83=Q N11
84=How is the damage of communication facility?  { se, co, sl, no, uk }
85=Q-$$$     this is the end mark of file-quest accessed by speril.
```

# STRUCTURAL ENGINEERING TECHNICAL REPORTS

80-8 "A Rule-Inference Method for Damage Assessment of Existing Structures", by M. Ishizuka, K. S. Fu and J. T. P. Yao.

80-9 "Nonlinear Hyperelastic (Green) Constitutive Models for Soils", by A. F. Saleeb and W. F. Chen (PB81-120032).

80-10 "Performance of Low-Rise Buildings -- Existing and New", by J. T. P. Yao.

80-11 "Application of Fuzzy Sets in Earthquake Engineering", by K. S. Fu and J. T. P. Yao.

80-12 "Plasticity Models for Soils", by E. Mizuno and W. F. Chen (PB81-120065).

80-13 "Effect of Human Errors to Structural Reliability", by B. Randich and J. T. P. Yao.

80-14 "A Plastic-Fracture Model for Concrete - Part I: Theory", by S. Hsieh, E. C. Ting and W. F. Chen.

80-15 "Analysis of Soil Response with Different Plasticity Models", by E. Mizuno and W. F. Chen.

80-16 "Upper Bound Limit Analysis of the Stability of a Seismic-Infirmed Earthslope", by S. W. Chan, S. L. Koh and W. F. Chen (PB81-117178)

80-17 "Inference Method for Damage Assessment System of Existing Structures", by M. Ishizuka, K. S. Fu and J. T. P. Yao.

80-18 "Data Analyses for Safety Evaluation of Existing Structures", by S. J. Hong Chen and J. T. P. Yao (PB81-186074).

80-19 "Identification of Hysteretic Behavior for Existing Structures", by S. Toussi and J. T. P. Yao (PB81-178113).

80-20 "Influence of Small End Restraint on Strength of Wide-Flange Columns", by H. Sugimoto and W. F. Chen.

80-21 "Curved Bridge Response to A Moving Vehicle", by J. Genin, E. C. Ting and Z. Vafa.

80-22 "A Unified Numerical Approach for Thermal Stress Waves", by E. C. Ting and H. C. Chen.

80-23 "Structural Impedance Method for Transient Bridge Response Subjected to General Traffic Conditions", E. C. Ting and R. Mirghaderi.

80-24 "Inelastic Cyclic Behavior of Tubular Members", by S. Toma and W. F. Chen.

80-25 "Unified Finite Element Approach for Coupled Thermal Stress Waves", by E. C. Ting and G. A. Keramidas.

81-1 "Effect of Small End Restraint on Strength of H-Columns", by H. Sugimoto and W. F. Chen.

81-2 "Limit Analysis of Lateral Earth Pressures on Rigid Walls Retaining Cohesionless Soils", by M. F. Chang and W. F. Chen.

81-3 "NFEAP - General Description, Sample Problems and User's Manual (1980 Version)", by S. S. Hsieh, E. C. Ting and W. F. Chen.

81-4 "Evaluation of Seismic Factor of Safety of a Submarine Slope by Limit Analysis", by C. J. Chang, W. F. Chen and J. T. P. Yao

81-5 "Inexact Inference for Rule-Based Damage Assessment of Existing Structures", by M. Ishizuka, K. S. Fu, and J. T. P. Yao.

81-6 "Theoretical Treatment of Certainty Factor in Production Systems", by M. Ishizuka, K. S. Fu, and J. T. P. Yao.

81-7 "Dynamic Stability of Curved Flow-Conveying Pipes", by E. C. Ting and Yi-Chen Liu.

81-8 "Cyclic Behavior of Tubular Sections", by S. Toma and W. F. Chen

81-9 "Structural Identification Control and Reliability in Wind Engineering Research", by J. T. P. Yao.

81-10 "Damage Assessment and Reliability of Existing Buildings", by J. T. P. Yao.

81-11 "Bibliography on Folded Plates (Theory, Material and Construction)" by C. D. Sutton and M. R. Resheidat

81-12 "NFEAP - General Description, Sample Problems and User's Manual (1980 Version) Part II", by S. S. Hsieh, E. C. Ting and W. F. Chen.

81-13 "Recent Advances on Analysis and Design of Steel Beam-Columns in USA", by W. F. Chen.

81-14 "Assessment of Seismic Displacements of a Submarine Slope by Limit Analysis", by C. J. Chang, W. F. Chen and J. T. P. Yao.

81-15 "Hystereses Identification of Multi-Story Buildings", by S. Toussi and J. T. P. Yao

81-16 "Inelastic Cyclic Analysis of Pin-Ended Tubes", by S. Toma and W. F. Chen.

81-17 "Cyclic Analysis of Fix-Ended Steel Beam-Columns", by S. Toma and W. F. Chen.

81-18 "Response of Truss Bridge to Traveling Vehicle", by E. C. Ting and J. Genin.

81-19 "Identification of Structural Characteristics Using Test Data and Inspection Results", by J. T. P. Yao

81-20 "Lateral Earth Pressures on Rigid Retaining Walls Subjected to Earthquake Forces", by M. F. Chang and W. F. Chen.

81-21 "Constitutive Relations and Failure Theories (Chapter 2)", by W. F. Chen (Chairman), Z. P. Bazant, O. Buyukozturk, T. Y. Chang, D. Darwin, T. C. Y. Liu and K. J. Willam.

81-22 "A Numerical Approach For Flow-Induced Vibration of Pipe Structures", by E. C. Ting and A. Hosseinipour.

81-23 "Seismic Safety Analysis of Submarine Slopes", by C. J. Chang, W. F. Chen and J. T. P. Yao.

81-24 "Inference Procedure with Uncertainty For Problem Reduction Method", by M. Ishizuka, K. S. Fu and J. T. P. Yao.

81-25 "Serviceability and Reliability of Antenna Structures in Part I: Theory", S. H. Wang, J. T. P. Yao and W. F. Chen.

81-26 "Fuzzy Statistics and Its Potential Applications in Civil Engineering", by J. T. P. Yao, K. S. Fu, M. Ishizuka.

81-27 "A Rule - Inference Method for Damage Assessment", by M. Ishizuka, K. S. Fu, and J. T. P. Yao.

81-28 "Lateral Load Capacity of Structural Tee X-Bracing", by A. D. M. Lewis and K. Nematollaahi.

81-29 "Stability of Structural Members with Time Dependent Material Properties", by E. C. Ting and W. F. Chen

81-30 "Response of Plate Bridges to a Moving Mass", by J. Genin, E. C. Ting & Mehdi Ilkhani-Pour

81-31 "Probabilistic Methods for the Evaluation of Seismic Damage Of Existing Structures", by J. T. P. Yao

81-32 "Lok-Test - A Non-Destructive Concrete Compressive Test", by S. Mofid, W. F. Chen, M. J. Gutzwiller

81-33 "Cyclic Inelastic Behavior of Steel Tubular Beam-Columns", by D. J. Han and W. F. Chen