DEVELOPING AND TRACING EXPERT SYSTEMS

Wm. Randolph Franklin
Rabul Banaal

Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, NY, 12180, USA

Abstract

We describe the Expert System Parsing Environment (ESPE), a system for debugging and testing expert systems written in IBM's Expert System Development Environment (ESDE). ESPE can perform the following operations: count the number of paths between each pair of nodes (parameter-value combinations or rules), split the expert system by focus control block (FCB) and draw the relationships in each section, and perform a sensitivity analysis. Here, the values of any output parameters that might change when a given input parameter changes from one specified value to another are listed. Also input parameters whose value change might cause a given output change are given. ESPE gives the expert system designer some of the software engineering tools used in traditional programming languages.

Introduction

An expert system is an expensive term for a program that is written in an expert system shell, which is a programming language, and often a badly designed one from a software engineering viewpoint. Many shells lack concepts of information hiding and modularity. Their mix of forward reasoning, backward reasoning, fuzziness, etc reminds one of PL/I, which had so many overlapping features that their interactions were not humanly predictable in advance. For example, the PL/I statement

\[ X = 25 + 1 / 3 ; \]

would set \( X \) to 5.333333 and raise an fixed overflow condition that was usually disabled. As larger expert systems are written, we may expect similar counterintuitive results, and therefore there is a need for systems to debug and test expert systems.

Software Engineering Review

Since, as expert systems get larger, issues of debugging and testing will become more important, it is reasonable to review the traditional software engineering life cycle, which has several stages.

Requirements Analysis
Specification
Design
Coding
Debugging and Testing
Maintenance

During requirements analysis the broad parameters for solving a problem are considered. For example, if the problem is to pay a company's employees, then we might keep longhand accounts, hire a payroll processing company, buy a computer and software, or buy a computer and write our own software.

In the specification stage we decide user visible parameters such as whether to handle both semi-monthly and biweekly employees, and the length of the name field.
The actual routines and their interfaces such as common blocks and argument lists are part of the design process. Next we code the system and produce something without syntax errors. In a student term project, this stage is often reached the night before the project is due.

Next we test the project with data to identify the system to the customer.

Finally throughout the system’s life there is continual maintenance. It may be due to uncovered bugs, new hardware, or customer requests for enhancements.

Experience with large projects has shown that the first four stages (requirements analysis, specification, design, and coding) together consume less than 25% of the system’s total cost and time. Debugging and testing require another 25%, while maintenance after delivery costs as much as everything before combined.

However, much current research in expert systems is concerned with issues such as expert system shell, that is language, design. This research, which affects only the coding part of the life cycle process, is about where software engineering was in the 1960s. Some of the few references on debugging expert systems or on graphical access to expert systems are 4,7,8,9. There are many references to expert systems that debug hardware, such as 3,6. Prolog has been used to implement expert systems, as described in 5.

1. ESPE counts the number of paths, of any length, between every pair of nodes and prints a table of the information.
2. ESPE splits the expert system into pieces determined by the hierarchy of the Focus Control Blocks.
   a) For each part, it produces a plot showing which nodes are joined by paths.
   b) It prompts the user for a key word, and determines all the parameters that contain that word in their name or prompt.
3. ESPE performs a sensitivity analysis of the expert system. Consider some input parameter.
   a) For each pair of values that that parameter may hold, ESPE lists the output parameter-value pairs that may be produced from only the first input value, from only the second, or from both.
   b) ESPE also allows the user to specify pairs of values for any output parameter and lists the input parameters whose value changes could cause that output change.

Parsing The Expert System

The expert systems we analyze are in the form of ESDE EXPORTED files as shown by these examples of rule texts. For an animal identification game, we have rules such as shown in figure 1.

```plaintext
@RULE GIRAFFE2
@PROP Rule text
If animal_class is 'mammal' and
animal_appearance is 'has long neck' and
animal_color is 'dark spots'
then animal = 'giraffe'
```

Figure 1: Animal Expert System Rule

In a system to debug hardware, we have rules such as figure 2.

The parsing process proceeds as follows:
1. Preprocess the expert system to remove a few problems such as unbalanced quotes and con-
the rule text, and the parameters read and written by each rule. Examples of these facts are as follows.

\text{nom} \ (\text{rule}, a13). \\
\text{prop} \ (\text{rule}, a13, [\text{rule}, \text{text}, \text{if}, \text{faulttwo}, ...]). \\
\text{reads\_param} \ (a13, \text{faulttwo}_1). \\
\text{writes\_param} \ (a13, \text{ps2x}).

This process abstracts away the internal structure of the rule and ignores how the rule combines its inputs values to determine its output. We have another program that considers the internal structure and plots an and-or graph of exactly how each rule reads its parameters and values. With this more complete information describing each rule, we could, if desired, simulate most of the expert system by converting the rules into Prolog clauses and ignoring the expert system shell. This would make the expert system runnable very efficiently on the special purpose Prolog microcomputers that are now appearing.

5. Convert the expert system into a directed acyclic graph and assert the relevant nodes and arcs. Each parameter-value pair and each rule becomes a node. Each node has a text string label associated with it that is used as a label elsewhere in the program. The correspondence between the parameter-value pairs or rules and their labels are recorded in the facts \text{par\_label} and \text{rule\_label}. The labels of parameter and rule nodes start with \text{P} and \text{R} respectively so that all the parameter nodes will appear together after sorting. Some sample facts are:

\text{rule\_label}(A13, 'R: A13'). \\
\text{par\_label}(\text{PS2X}, \text{yes}, 'P: PS2X = yes'). \\
\text{node}('P: PS2X = yes'). \\
\text{node}('R: A13').

Then each instance of a rule reading or writing a parameter with some value becomes an arc thus:

\text{arc}(\text{R: A13}, 'P: PS2X = yes').
6. Assert certain identification facts:

   filename(P128).
   title(AESOP).

7. Derive certain facts, such as input nodes, which should be supplied by the user, output nodes, which should be read by the user, and unused nodes.
Path Counts On The Directed Graph

We count the number of paths between each pair of nodes in the graph and display the information, as shown for the animal expert system, in Figure 3. Only the nonzero rows and columns are listed. Thus parameter Animal is never used by any later parameter or rule (it is the user-visible result). Animal Appearance is used many times, but unevenly. For instance, Animal Appearance = has claw is used 4 times while Animal Appearance = has feathers is used on 14 different paths. Since Animal Appearance = has hooves is used only twice, we might say that it is not covered very completely by this expert system.

The table of counts for large expert systems is split and printed in strips.

Although this table has a superficial appearance to a low level core dump, that appearance is misleading since it actually results from considerable processing and the abstraction of the relevant information from a large mass of raw data. The path counts are determined by a Prolog program that operates iteratively. Initially all arcs are paths of length one. At any stage, the paths of length K are determined by catenating all paths of length K-1 to all adjacent arcs and then asserted. This requires time at worst proportional to the number of paths at each length. Any cycles are detected and arbitrarily broken. The output is done in PL/1 because of its more flexible format control.

Partitioning By Focus Control Block

A focus control block (FCB) is a means of grouping the parameters and rules of a large expert system so that certain rules do not become active or usable until certain conditions are reached. The focus control blocks, which form a tree structure, are a convenient way to display only part of a larger expert system. For each leaf FCB we can produce a plot of all the arcs that have at least one node in that FCB hierarchy, the set of all FCBs and its ancestors up to the root. This information can be displayed on either a 3250 vector refresh CRT, or on a 3277 Graphics Attachment Storage CRT.

Since parameters can have short, domain specific names that are not obvious to users, the program can prompt the user for a string, and list all the parameters that contain it in their prompt.

Sensitivity Analysis

It is useful to know what different goals are reached depending on different values in the input parameters. We take each pair of possible values of an output parameter, such as Animal, and determine what different values of the input parameters might cause this distinction, as shown in figure 4.

PARAMETER NAME? animal
PARAMETER ANIMAL VALUES 'albatross' AND 'penguin' ARE DISTINGUISHED BY:
(1) ANIMAL LOCOMOTION = flies
(2) ANIMAL LOCOMOTION = flies well
(2) ANIMAL LOCOMOTION = swims

PARAMETER ANIMAL VALUES 'albatross' AND 'ostrich' ARE DISTINGUISHED BY:
(1) ANIMAL LOCOMOTION = flies
(1) ANIMAL LOCOMOTION = flies well
(2) ANIMAL APPEARANCE = has long neck

PARAMETER ANIMAL VALUES 'albatross' AND 'giraffe' ARE DISTINGUISHED BY:
(1) ANIMAL CLASS = bird
(1) ANIMAL LOCOMOTION = flies
(1) ANIMAL COLOR = black and white
(2) ANIMAL CLASS = mammal
(2) ANIMAL APPEARANCE = has long neck
(2) ANIMAL COLOR = dark spots

Figure 4: Input Parameter Values That Distinguish Between Output Parameter Values in the Animal Expert System.

A (1) means that this value can lead to the first output value of the pair; and a (2) labels values that cause the second. Here we see that the only input difference between albatross and penguin is in values of the Animal Locomotion parameter, and that there are a total of three differences.
**Albatross and ostrich** are a little more different. Although there are still only three differences, they involve two parameters: *Animal_Locomotion* and *Animal_Appearance*. Albatross and giraffe have seven differences involving four different input parameters, which is reasonable. If the number of differences detected does not accord with the user’s view of the problem, then we have uncovered an area of weak coverage in the expert system.

A part of the output from the analysis of a larger, hardware-debugging expert system is given in figure 5.

**Figure 5: Input Parameter Values That Distinguish Between Output Parameter Values in the Hardware-Debugging Expert System**

The above method may generate a large number of pairs of values for a complex output parameter. Therefore we also allow the user to specify two pairs of values and see just that comparison.

Since the particular input parameter values are not as important as the fact that those parameters are involved, we also create a table showing, for each pair of output parameter values, which input parameters might distinguish between them. See figure 6 for the animal expert system. Looking down the columns, this shows that three value pairs: *albatross-penguin*, *giraffe-cheetah*, and *tiger-cheetah* are distinguished by only one input parameter each. The surprising fact that only difference between giraffe and cheetah is *animal_appearance* suggests that this expert system might be improved here.

Looking across the rows, we see that *animal_locmotion* distinguishes only 11 pairs of output values, while *animal_appearance* is more important since it distinguishes 18 pairs. This information is automatically summarized by ESPE in figure 7.

A sample output from analyzing a larger hardware-debugging expert system is this shown in figure 8.

Here we are analyzing input parameter *P3byp1* which has four values to see how it affect output parameter *End*, which has three values. For each pair of possible input values, we see which of the pair might influence each output value. For example, neither of the pair *Install or relocate or EC/MES Activity* can cause any of those output values. Of the pair *Preventive Maintenance or Service Call*, the first might cause only the first output value, while the second input might cause the second or third output values.

Once of the applications of this work is to judge whether an uncertain input is important is making a particular diagnosis. Thus the first pair of possible inputs is useless in discriminating between the three listed outputs, and only the last pair can give any of the three of the outputs depending on which one of the pair is picked.

**Summary**

We have applied certain software engineering techniques to the analysis of expert systems. We convert the expert system to a directed acyclic graph and then count paths and perform sensitivity analyses. This allows the user to test the completeness of the expert system's coverage of the problem.
DIFFERENCES IN VALUE FOR: ANIMAL
NUMBER OF VALUE PAIRS: 21
NUMBER OF PARAMETERS USED TO MAKE DISTINCTIONS: 4

VALUE PAIRS:
1 albatross penguin
2 albatross ostrich
3 albatross zebra
4 albatross giraffe
5 albatross tiger
6 albatross cheetah
7 penguin ostrich
8 penguin zebra
9 penguin giraffe
10 penguin tiger
11 penguin cheetah
12 ostrich zebra
13 ostrich giraffe
14 ostrich tiger
15 ostrich cheetah
16 zebra giraffe
17 zebra tiger
18 zebra cheetah
19 giraffe tiger
20 giraffe cheetah
21 tiger cheetah

PARAMETERS:
1 ANIMAL_LOCOMOTION
2 ANIMAL_APPEARANCE
3 ANIMAL_CLASS
4 ANIMAL_COLOR

LAST COLUMN INDICATES NUMBER VALUE PAIRS THIS PARAMETER DISTINGUISHES BETWEEN
LAST ROW INDICATES NUMBER PARAMETERS DISTINGUISH BETWEEN EACH VALUE PAIR.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOT: 1 2 4 4 4 4 4 4 4 3 2 3 2 2 2 2 1 1

Figure 6: Input Parameters That Distinguish Between Output Parameter Values in the Animal Expert System System

Acknowledgements
This work was supported by the National Science Foundation under FY1 grant no. DMC-8351942, and by the Data Systems Division of the International Business Machines Corporation.
There were 21 pairs of values, and 4 parameters distinguish values.

The average number of distinguishing characteristics between goal value pairs is 2.76.

Value pairs with fewer than 2 differences:
- albatross, penguin -- 1 difference
- giraffe, cheetah -- 1 difference
- tiger, cheetah -- 1 difference

Parameters that distinguish more than 15 value pairs:
- ANIMAL_APPEARANCE -- used for 18 differences.
- ANIMAL_COLOR  -- used for 17 differences.

Figure 7: Sensitivity Analysis of the Animal Expert System

Comparison of conclusions reachable by parameter PRDTYPEI values.

Conclusions examined are END.

Input parameter value pairs:
1. Install or relocate \- EC/MES Activity
2. Install or relocate \- Preventive Maintenance
3. Install or relocate \- Service Call
4. EC/MES Activity \- Preventive Maintenance
5. EC/MES Activity \- Service Call
6. Preventive Maintenance \- Service Call

Goal values:
1. INSEE THE FOLLOWING DOCUMENTATION FOR PM
2. FOLLOW VOLUME B01, PCE START MAP
3. REFER TO IOPD SECTION OF VOL A02

Array rows represent input value pairs.
Array columns represent goal values.

Array contents have the following meaning:
A--the goal value is reachable only from the first input value.
B--the goal value is reachable only from the second input value.
C--the goal value is reachable from both values in the input pair.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8: Comparison of Reachable Conclusions
References


