1989

An expert system for ultrasonic flaw classification

Stephen M. Nugen
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd
Part of the Computer Engineering Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/239
An expert system for ultrasonic flaw classification

by

Stephen Mark Nugen

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Department: Electrical Engineering and Computer Engineering
Major: Computer Engineering

Approved:

Signature redacted for privacy

Iowa State University
Ames, Iowa
1989
# TABLE OF CONTENTS

1 INTRODUCTION ........................................................................... 1  
1.1 Problem ........................................................................ 1  
1.2 Relevant Artificial Intelligence Issues ................................. 3  
1.3 Results and Contributions .................................................. 5  
1.4 Structure of the Thesis ......................................................... 5  

2 PROBLEM DESCRIPTION ............................................................. 7  
2.1 Context ............................................................................. 7  
2.2 Specific Problem Description ............................................. 8  

3 SOLUTION DESCRIPTION .......................................................... 12  
3.1 Introduction ....................................................................... 12  
3.2 FEAP Feature Evaluation .................................................. 18  
3.3 Flaw Response Features .................................................... 20  
3.4 Human Expertise ............................................................... 23  
3.5 External FLAP Rules ........................................................ 25  
3.6 Internal FLAP Rules ......................................................... 36  
3.7 FLAP Rule Translator ....................................................... 39  
3.8 FLAP Inference Engine ...................................................... 45  
3.9 Flaw Type Results ............................................................. 59  
3.10 FLAP Explanation Facility .............................................. 61  
3.11 Summary of Results ......................................................... 63
1 INTRODUCTION

This thesis describes the results of a research program which focused on the use of artificial intelligence techniques to solve a problem in the domain of nondestructive evaluation (NDE). The work was performed at the Center for Nondestructive Evaluation at Iowa State University under the supervision of Dr. Charles Wright, Department of Electrical Engineering and Computer Engineering, and Dr. Lester Schmerr, Department of Engineering Science and Mechanics.

1.1 Problem

Nondestructive evaluation (NDE) engineers use a variety of techniques to detect flaws as anomalies in the material being examined. One of the most widely used techniques is to direct ultrasonic sound energy into the material and then analyze the energy which is reflected back from discontinuities such as surface flaws and embedded flaws. The analysis of this backscatter energy is typically performed by a human specialist who acquires his or her skill through formal education, extensive on-the-job training, and experience. Different levels of expertise are recognized by industry with a 3-level certification process administered by the American Society for Nondestructive Testing (ASNT).

Because nondestructive testing generates such large amounts of raw data, its
analysis tends to be an arduous, repetitive task which is prone to error. Accordingly, the industrial sponsors of the Center for NDE have asked us to investigate techniques which will automate the evaluation process.

The knowledge used by human engineers in their evaluation can be divided into two overlapping areas. The theoretical component uses basic principles from physics and a knowledge of the material properties to predict the interaction between the sound waves and flaws. The practical or experimental component is often expressed as "rules-of-thumb" or heuristic knowledge. Models have been constructed to automate the application of theoretical knowledge, but their usefulness is compromised by the necessary simplifying assumptions and the computational costs associated with complex algorithms. Adaptive learning networks have been constructed [13, 26, 27] that attempt to learn rules that can be used for flaw evaluation, but these rules are specific to particular materials, geometries, and test bed configurations. Additionally, adaptive learning schemes suffer from the lack of an explanation facility [15] which is needed for user acceptance of automated evaluation systems.

To overcome these drawbacks of model-based and adaptive learning automation schemes, the industrial sponsors requested a program of research to investigate new approaches based on the techniques of artificial intelligence. The specific problem chosen for the feasibility study was to classify the flaw type, that is, distinguish crack-like flaws from volumetric flaws. The goal of the research was to demonstrate feasibility by the construction of an expert flaw classification system which could then be used as a starting point for the construction of similar expert systems by the sponsors aimed at their specific NDE concerns. This goal is in concert with the
technology transfer mission of the Center for NDE.

1.2 Relevant Artificial Intelligence Issues

Artificial intelligence (AI) is the study of how to make computers perform tasks which, at the moment, people perform better [17, 19, 29]. Two major subsets of AI are neural networks and expert systems.

An expert system is a computer program which uses knowledge and inference procedures to solve problems difficult enough to require significant human expertise for their solution. Construction of such a program is justified for well-defined problems where the human expertise is scarce and a method exists for confirming the correctness of the implementation [10]. Three open areas in expert systems are knowledge representation, handling uncertainty, and inference strategies.

The two most prevalent schemes for knowledge representation are semantic networks and rules. Semantic networks are collections of nodes organized by the links between them [7]. The nodes in such systems may be frames which contain both data and procedures [1, 8, 14]. The network may be organized into a hierarchical system which supports the concept of inheritance. Rules have two components: the antecedent or "if" part and the consequent or "then" part [11]. While semantic networks have the potential to capture a much richer body of knowledge, rules excel in their ability to capture knowledge in an explicit form which is easily understood and verified. For this reason, rule-based expert systems, also known as production systems, are seeing wide
acceptance for industrial-strength problems.

Human knowledge, especially heuristic knowledge, is oftentimes expressed using language phrases like "probably", "almost", "usually", and so forth. Three well-established methods for representing this uncertainty are probability theory [1], confidence factors [2, 23, 24], and fuzzy logic [6, 32]. When facts are established by more than one rule, the uncertainties must be blended. Also, uncertainty must be propagated from one step to the next just as measurement uncertainty is propagated from one calculation to the next [31].

Classic inference strategies fall into two categories: backward or goal-directed and forward or data-driven. Goal-directed systems start with the goal and try to find rules whose consequent or conclusion is the goal. Data-driven systems fire rules to reach a goal which is not known beforehand. The reasoning in these systems can be either monotonic where no conclusions are ever retracted or nonmonotonic.

The problem described in subsection 1.1 is an appropriate one for studying artificial intelligence (AI) issues since it is well-defined, the human expertise is in short supply, the difference between expert and nonexpert performance is clear, and an oracle is available to verify results. The particular issues studied in this research are rule-based knowledge representation, a hybrid inference scheme, and user interfaces for expert system.
1.3 Results and Contributions

An expert system for flaw type classification has been designed, implemented, and tested. In doing so, I have introduced the techniques of artificial intelligence, particularly those of expert systems, to a new problem domain: nondestructive evaluation. These techniques deliver one more tool to the collection of methods used by NDE engineers to evaluate critical structures such as aircraft wings and coolant tubes in nuclear power plants.

I have augmented existing techniques for the construction of rule-based expert systems in three areas. (1) I have demonstrated the feasibility of bi-level knowledge representation by constructing a software tool which compiles rules from an English-like form easily understood by humans into a LISP-like form easily interpreted by the expert system run-time environment. (2) I combined elements of forward and backward inference to design a simple conclusion mechanism which works in the forward direction toward fixed goals in a manner analogous to courtroom procedures. (3) I have demonstrated a useful user interface in presenting the results of an expert system to developers and users.

1.4 Structure of the Thesis

The research described in this thesis is part of a larger project whose results have already been presented in part at the 1986, 1987, and 1988 conferences, Review of
Progress in Quantitative Nondestructive Evaluation [18, 20, 21]. One of the main goals of this larger project was to construct a complete flaw classification system using AI software tools. This system, called FLEX (for FLaw EXpert), has been developed over the last three years at the Center for NDE. FLEX consists of two cooperating programs, FEAP (for FEAture Processing) and FLAP (for FLAww Processing).

FEAP's task is to extract from a set of flaw response measurements those features which are useful for the classification task. This is accomplished by evaluating a set of decision trees (which define these features) using ideas from fuzzy set and fuzzy logic theory. The details of FEAP are given by Ken Christensen in his thesis [4].

FLAP uses the features defined by FEAP to decide if the flaw type is crack-like or volumetric. This is accomplished through a rule-based expert system approach. The details of FLAP are given in this thesis.

The remainder of this thesis is structured into a series of major sections which describe the specific problem, its solution, the results of applying the solution to the problem, and the conclusions which follow. In each section, the strategy is to introduce the topics in summary fashion and then elaborate on them further in the text. The bulk of the research is described in the solution section which is organized by the 11 functional elements of the solution. Figures are placed within the discussion. A bibliography appears at the end.
2 PROBLEM DESCRIPTION

This section describes the target of the research in terms of its context and the specific problem.

2.1 Context

Nondestructive evaluation seeks to assess the fitness of structures without destroying them in the process. For many structures, the assessment is meant to answer the question: will the component fail while in-service? This evaluation may be periodic as in the case of regularly scheduled inspections or by exception when some event raises the suspicion of failure. NDE professionals are recruited from a variety of disciplines such as engineering mechanics, materials science, physics, and metallurgy with support from disciplines such as computer engineering, electrical engineering, and computer science.

The structural fitness of an object can be compromised by flaws in the object such as cracks, voids, and inclusions. These flaws may be surface breaking or completely embedded. Over time, a collection of methods have been developed to detect these flaws and characterize them with regard to type, size, and orientation. These methods include visual inspection, ultrasonic, radiographic, magnetic, eddy current, penetrants, and thermographic techniques [9].
The technique selected for this research was the ultrasonic method. In this method, electrical energy from a pulser is converted into mechanical energy, sound waves, by a transducer and propagated into the part under test. The experimenter can either place the transducer directly on the surface of the part (contact testing) or use a intermediate medium, such as a fluid, to couple the sound energy into the part (immersion testing). Figure 2.1 shows a typical ultrasonic immersion set up. Part of the incident energy is reflected back to the transducer at points of discontinuity such as front surface, back surface, and flaws. The reflected energy is converted to electrical energy by the transducer and sent to the receiver for measurement. The energy level of the reflections over time constitutes a time domain trace. A Fast Fourier Transform was used to translate the time domain trace to the corresponding frequency domain trace. Both time and frequency domain data were used in this research and will be referred to collectively as the flaw response.

2.2 Specific Problem Description

The system constructed is designed to distinguish crack-like flaws from volumetric flaws using ultrasonic flaw response data. Thus, our system is a classification system. Our reasons for focusing on the flaw type characterization are as follows:

1. Knowledge of flaw type is, in itself, directly useful in reliability assessments. This is because cracks are usually more dangerous to structural integrity than volumetric flaws and because cracks tend to propagate under stress over time while
Figure 2.1 Ultrasound Immersion Testing
volumetric flaws do not.

2. Current flaw sizing techniques use different algorithms for crack flaws versus volumetric flaws [3]. Thus, the flaw type must be known prior to employing these algorithms. Furthermore, if the flaw type is known to be crack, additional processing can be used by the sizing function to systematically eliminate some of the noise. This noise elimination significantly improves the accuracy of the sizing [22]. Figure 2.2 shows how flaw type determination fits into the flaw characterization decision tree.

3. The expertise needed for flaw classification is readily available at the Center for NDE.

4. The data needed to test solutions are available at the Center. Some of the data sets are generated from models while other sets are the results of experimental ultrasonic scans performed at the Center.

5. Flaw type classification is a small, well-defined problem which lends itself to an expert system solution.
Figure 2.2. Flaw Characterization Decision Tree
3 SOLUTION DESCRIPTION

This section describes our solution to the flaw type determination problem.

3.1 Introduction

The solution had to meet the following criteria:

1. Demonstrate the feasibility or infeasibility of applying artificial intelligence techniques to the specific NDE problem.

2. Automate the decision making process.

3. Keep the decision making knowledge explicit so that it can be easily comprehended by the human experts.

4. Justify the solutions in a manner amenable to human verification of correctness.

5. Serve as a starting point for industrial users to develop their own custom solutions.

The approach taken was to construct a software package consisting of two cooperating intelligent systems along with signal processing and control programs. The entire system was given the name FLEX for FLaw EXPert. Figure 3.1 shows the
FLEX operating environment.

The software was originally developed on a Symbolics 3670 LISP computer with a bit-mapped graphics (1100 X 700) monochrome display, mouse, 6 MB of main memory, and 474 MB of file storage and swap space. The 3670 employs a tagged architecture and special microcode optimized for LISP execution. The operating system is Genera 7.1.

As part of the Center's technology transfer mission, the software has been ported to a Apple Macintosh II microcomputer (68020 CPU and 6881 FPU) with bit-mapped graphics (640 X 480) 256-color display, 5 MB of memory, and 80 MB file storage. The LISP [25] Compiler is Allegro Common LISP version 1.2 and the operating system is Apple System version 6.02. FLEX execution times on the Macintosh are on the order of two to three times slower than execution times on the Symbolics.

Figure 3.2 shows the FLEX system architecture. The two intelligent systems are named FEAP for FEAture Processing and FLAP for FLAwh Processing. These two systems are loosely coupled [16]. The output conclusions of FEAP become the input facts of FLAP. This thesis concentrates on the FLAP portion of FLEX.

Because FLAP was designed as a rule-based expert system, it contains the basic components given in Figure 3.3 [10, 11]. The manner in which this simple model is
Figure 3.1 FLEX Operating Environment
Figure 3.2 FLEX System Architecture
Figure 3.3 Basic Components of an Expert System
Figure 3.4. FLAP System Architecture
explicitly embedded in the overall FLEX architecture is shown in Figure 3.4.

3.2 FEAP Feature Evaluation

FEAP is primarily the creation of Ken Christensen and is described by his thesis [4]. The description of it here is limited to what functions it performs rather than how it performs them.

FEAP's responsibility is to extract from the flaw response a set of beliefs that the response contains certain features. FLAP then makes use of those feature evaluations, along with its knowledge of which features are associated with which flaw types, to reach a conclusion.

The FEAP process can be viewed as a mapping from the signal amplitudes of the flaw response to a set of features as shown in Figure 3.5. Each $F_i$ represents a single feature. Currently, FEAP evaluates 9 features corresponding to 9 FLAP rules. Additional rules and features are under consideration. While each signal is evaluated independently from every other signal, the evaluation of $F_i$ may be dependent on the evaluation of $F_j$. Each $F_i$ has four attributes associated with it as follows:

1. $\text{Val}(F)$: a real-valued function in the range $[-1,+1]$ representing the confidence that feature $F$ is present in the flaw response signal. Confidence factors are discussed in subsection 3.5.
Figure 3.5 FEAP Mapping
2. **Name(F):** a LISP symbol-name [30] that uniquely identifies the feature. Name(F) is currently chosen from the set \{FLASH\_PTS, RINGING, CREEP\_WV, RAYLEIGH\_WV, NORMAL\_INCI, DEEP\_NULLS, SHALLOW\_NULLS, SHARP\_NULLS\}. This attribute is used to construct the object-value tuples which are passed from FEAP to FLAP.

3. **Method(F):** algorithms and coefficients used by FEAP to calculate the value attribute. Notions from fuzzy set theory are used in this calculation.

4. **Type\_Assoc(F):** this attribute is maintained by FLAP to associate certain features with flaw type.

### 3.3 Flaw Response Features

Typically, a complete ultrasonic scan consists of measuring the reflected sound energy from different transducer orientations relative to a flaw. The angle between the receiving transducer and the flaw is called the **look angle**. A set of flaw responses consists of several data sets for a single flaw, each of them at a different look angle. FEAP evaluates the data for each angle independently of all other angles by constructing a set of tuples of feature evaluations, one for each look angle. While FLAP does not use this angle information directly, it will strengthen its conclusion when the features in one look angle tend to agree with the features of the other look angles. Likewise, FLAP's conclusions are weakened if FEAP finds different features
for different look angles.

The output of FEAP is stored as a text file for input by FLAP. By choosing to use a text file for this interface, feature evaluation files can be created with a text editor to test FLAP scenarios without the need to execute FEAP. Because FLAP was implemented before FEAP, this property was used for all of the initial FLAP tests. Additionally, keeping the data interchange in text format eases the task of verifying the program at crucial inspection points.

The text file contents consist of a single LISP list expression specified by the following BNF [12] grammar:

```
FLAW-Resp-Form ::= (Angle-1-Form Angle-2-Form Angle-m-Form)
             {where m = number of look angles available
              for this data set}

.Angle-*.Form ::= (Angle-ID Name-Value-1
               Name-Value-2 ... Name-Value-n)
             {where n = number of features available for this look angle}

.Angle-ID ::= "<Data-Type-ID><Angle-Number>.<Data-Set-ID>"

<Data-Type-ID ::= <char><char><char>
```
<Angle-Number> ::= <digit>

<Data-Set-ID> ::= <char><char>

<Name-Value-> ::= (<Name> <Value>)

{Name} ::= POS_PULSE | FLASH_PTS | RINGING | CREEP_WV | RAYLEIGH_WV | NORMAL_INC | DEEP_NULLS | SHALLOW_NULLS | SHARP_NULLS

<Value> ::= <digit><digit> | -<digit><digit> |
+<digit><digit>

(The value is assumed positive in the absence of a leading sign.)

<char> ::= A..Z..a..z

<digit> ::= 0..9

The following fragment of a form illustrates the grammar.

("sca1.c01" (POS_PULSE -0.9) (FLASH_PTS 0.9) ...
SHARP_NULLS 0.0)) ("sca2.c01" (POS_PULSE -0.7) ...) ...
("sca9.c01" (POS_PULSE 0.1) ... (SHARP_NULLS 0.3)))
In this example, the scattering amplitudes for sample C01 (a crack) has 9 viewing angles. The total flaw response information of data set C01 has been reduced to 81 (9 features for 9 viewing angles) name-value pairs.

There are no assumptions made about the number of viewing angles in a given data set or about the number of feature evaluations available in any one look angle. The FLAP design adopts the convention that if a Name-Value pair is not available to match on a FLAP rule, then the feature confidence is taken to be zero, uncertain, so that the rule has no effect upon the conclusion.

The deliberately loose coupling between FEAP and FLAP along with the text specification of the data interchange is also meant to suggest the desirability of building expert systems as small, cohesive, independent modules instead as of large, tightly coupled systems. In this implementation, for example, FEAP or FLAP could be entirely rewritten (into a neural network based scheme for instance) without affecting the other.

3.4 Human Expertise

The domain knowledge for FLAP was obtained from humans who were experts at classifying flaw type by inspection of the ultrasonic flaw response.
3.4.1 Source

The domain knowledge needed for the construction and evaluation of an expert system for the problem came primarily from Dr. Lester Schmerr, a Professor in the Engineering Science and Mechanics department at Iowa State University. Dr. Schmerr teaches courses in NDE and is a principal investigator at the Center for NDE. Other staff members of the center that have contributed their expertise are Mr. Sam Wormley, Dr. Timothy Gray, and Dr. James Rose.

The Center's 18 industrial sponsors have also reviewed this work semiannually since January, 1987. Their comments have been useful in evaluating trial solutions to the problem.

3.4.2 Extraction

There were two categories of domain knowledge required. First, we had to know how an human expert distinguished between crack and volumetric flaws from an examination of the flaw response. When we asked this question, the answers were given in terms of signal features. This knowledge was refined into a set of rules where some features support a finding of crack type while others support a finding of volumetric type. The second category of knowledge needed was how to recognize the presence of the features themselves in the flaw response. The experts cautioned us that their rules of thumb had a certain amount of uncertainty implicit in them.
A minimum base of expertise in terms of definitions and terminology was obtained through informal lectures and interviews with the human experts. Then, a set of experts was collected and asked to evaluate, in a cooperative fashion, a set of 44 flaw responses from crack and volumetric flaws while detailed notes were taken about the results and the methods used to arrive at them. Special attention was paid to the cases where the experts initially disagreed in their interpretations of the data. These data sets and the human expert evaluations of them became the basis for the construction of the FLAP knowledge base.

3.4.3 Validation

We were able to fully verify the domain knowledge in this instance because we were using data sets for known flaws. The data sets examined in the cooperative evaluation were either generated by a model of a given flaw type or were the result of ultrasonic test scans of manufactured flaws with known type. Because we had this perfect knowledge of what our conclusions should be, there was no need to convene an independent body of experts to validate our domain knowledge sources.

3.5 External FLAP Rules

3.5.1 Introduction

The FLAP knowledge base is structured as a set of If-Then rules which are represented in two forms: external and internal. This section describes the external
The relationship between the external rules and the internal rules is analogous to the relationship between program source code and program object code. Like source code, external rules are created and modified with standard text editors. They are easy to read, but inefficient in execution. They must be translated. Management of multiple versions of the external rules can be eased by application of existing software configuration tools.

The goal of this representation is to make the codified knowledge explicit and easily understood by domain experts who are not necessarily familiar with computer languages and data structures. This goal is driven in turn by the desire to spur acceptance among NDE engineers and managers as well as the need to allow modification of the knowledge base by nonprogrammers.

Free-form English prose is chosen to be the ideal model of communication between the human expert and the computer knowledge base. This model is then constrained by the simplicity of the translator program responsible for translating external rules into internal rules. As the English prose is allowed to become more free-form, the complexity of the translator increases dramatically. Because the scope of this implementation did not include natural language processing issues, the freedom of the prose is severely restricted.
3.5.2 Evidence Type

All of the rules relate features detected in the flaw response signal to a particular conclusion in terms of assigning evidence that supports or discourages a finding that the flaw type is crack or volumetric. The evidence provided by each rule falls into one of three categories: sufficient, necessary, or indicative.

An example of sufficient evidence is the unique response in the frequency domain signal of a crack viewed at normal incidence. This response, linearly increasing amplitude, is so characteristic of a crack type of flaw that it stands alone in its support of the crack hypothesis. This type of evidence is nonaccumulating.

An example of necessary evidence is the requirement that the leading edge response in the time domain for a crack flaw be of negative polarity. If a positive polarity leading edge pulse is detected, negative accumulating evidence is recorded to discourage a finding of crack type.

Most evidence is indicative. For example, ringing (resonance) in the time domain indicates a volumetric flaw, but the absence of ringing does not indicate a crack flaw. This type of evidence is positive accumulating.
3.5.3 Measuring Uncertainty

Uncertainty is represented in terms of confidence factors developed by Shortliffe and Buchanan for the MYCIN project [24]. Confidence factors (CF) are not the same as probabilities. In particular, a CF of $X$ in hypothesis $A$ does not imply a CF of $1-X$ in hypothesis NOT $A$.

Given a Measure Of Belief (MOB) in the interval $[0,1]$ where 0 represents no belief and 1 represents complete belief and a Measure Of Disbelief (MOD) in the interval $[0,1]$ where 0 represents no disbelief and 1 represents complete disbelief, then the Confidence Factor (CF) is defined as: $CF = MOB - MOD$. CF then is in the interval $[-1,+1]$ where -1 represents absolute disbelief, 0 represents uncertainty, and +1 represents absolute belief.

For the external rule structure, a mapping is defined to partition the range $[-1,+1]$ into nine discrete classes, each of which has a corresponding English phrase which is used in the external rule structure. This mapping is defined in Table 3.1.

3.5.4 Syntax

The rules are stored in a file as lines of text. All lines up to and including the one, "--*Begin Translation*--" are ignored by the translator. This convention allows the inclusion of free-form comments directly in the file itself. There are no expectations made about the number of rules. A listing of the actual file containing external FLAP
rules is given in Figure 3.6.

An informal BNF grammar [12] scheme is used to specify the syntax of the external rules. Pairs of curly brackets and the text within them are comments to the reader rather

Table 2.1. Mapping Uncertainty to Confidence Factors

<table>
<thead>
<tr>
<th>Partition</th>
<th>English Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0 ≤ CF ≤ -0.8</td>
<td>Certain disbelief</td>
</tr>
<tr>
<td>-0.8 &lt; CF ≤ -0.6</td>
<td>Strong disbelief</td>
</tr>
<tr>
<td>-0.6 &lt; CF ≤ -0.4</td>
<td>Moderate disbelief</td>
</tr>
<tr>
<td>-0.4 &lt; CF &lt; -0.2</td>
<td>Weak disbelief</td>
</tr>
<tr>
<td>-0.2 ≤ CF ≤ +0.2</td>
<td>Uncertainty</td>
</tr>
<tr>
<td>+0.2 &lt; CF &lt; +0.4</td>
<td>Weak belief</td>
</tr>
<tr>
<td>+0.4 ≤ CF &lt; +0.6</td>
<td>Moderate belief</td>
</tr>
<tr>
<td>+0.6 ≤ CF &lt; +0.8</td>
<td>Strong belief</td>
</tr>
<tr>
<td>+0.8 ≤ CF ≤ +1.0</td>
<td>Certain belief</td>
</tr>
</tbody>
</table>
Version 2.0 - 3/17/87 - smn

History:

1. New document which supersedes concl-rules.text version 2.1. The rule numbers are unchanged.

2. These rules are in the external syntax specified by flak-ext-syntax.text.

3. For version 1.1, changed "%" to "percent" to ease the translation task.

4. For version 1.2, changed rule 401 "one percent" to "1 percent", changed rule 406 "reponse" to "response".

5. For version 2.0, removed all references to thresholds, i.e. percentages. From now on, a positive value for a feature CF implies detection of the feature and the rule will fire.

---*Begin Translation*---

(Rule 200
(if a positive pulse is detected in the leading edge response of the flaw signals in the time domain)
(then there is strong disbelief in the stand-alone evidence supporting a determination of crack flaw)
(rem ))

(Rule 202
(if flash points are detected in the leading edge response of the flaw signals in the time domain)
(then there is moderate belief in the accumulating evidence supporting a determination of crack flaw)
(rem ))

(Rule 204
(if ringing is detected in the trailing response of signals in the time domain)
(then there is weak belief in the accumulating evidence supporting a determination of volumetric flaw)
(rem aka resonance))

---*End Translation*---

Figure 3.6. External FLAP Rules
(Rule 206
(if creep wave is detected in the trailing response of the flaw signals in the time domain)
(then there is weak belief in the accumulating evidence supporting a determination of volumetric flaw)
(rem ))

(Rule 208
(if Rayleigh waves are detected in the trailing response of the flaw signals in the time domain)
(then there is weak belief in the accumulating evidence supporting a determination of crack flaw)
(rem ))

(Rule 402
(if normal incidence of a crack is detected in the response of the flaw signals in the frequency domain)
(then there is strong belief in the stand-alone evidence supporting a determination of crack flaw)
(rem looking here for a continuous increase in magnitude with increasing frequency))

(Rule 404
(if deep periodic nulls with decaying amplitude are detected in the response of the flaw signals in the frequency domain)
(then there is moderate belief in the accumulating evidence supporting a determination of crack flaw)
(rem deep nulls may also be indicative of non-spherical volumetric flaws - see also rule 202))

(Rule 406
(if shallow nulls resting on a relative plateau are detected in the response of the flaw signals in the frequency domain)
(then there is moderate belief in the accumulating evidence supporting a determination of volumetric flaw)
(rem ))

Figure 3.6 Continued.
(Rule 408
(if very sharp nulls are detected in the response of the
flaw signals in the frequency domain)
(then there is moderate belief in the accumulating evidence
supporting a determination of volumetric flaw)
(rem expected when ringing is present - see also rule 204))

Figure 3.6 Continued.
than part of the BNF grammar. The parentheses are mandatory—the translator expects each external rule to be a valid LISP form.

<FLAP-Rule> ::= 
   (Rule <Rule-Number>
   (if <Feature-Name> detected in the <Signal-Component>
   response of the flaw signals in the <Signal-Domain> domain)

   (then there is <Belief-Level> in the <Evidence-Type>
   evidence supporting a determination of <Conclusion-Type>)

   (rem <Remarks>))

<Rule-Number> ::= 
   200..299 {corresp. to features in the time domain}
   400..499 {corresp. to features in the frequency domain}
   500..599 {reserved for integrated time domain rules}
   600..699 {reserved for miscellaneous rules}
   800..899 {reserved for meta-rules, i.e., rules about rules}
<Feature-Name>::= 
  a positive pulse is 
  I flash points are 
  I ringing is 
  I creep wave is 
  I Rayleigh waves are 
  I normal incidence of a crack is 
  I deep, periodic nulls with decaying amplitude are 
  I shallow nulls resting on a relative plateau are 
  I very sharp nulls are 
  I small real parts are {not currently used} 
  I a step function is {not currently used} 

<Signal-Component>::= leading edge | trailing | <NIL> 

<Signal-Domain>::= time | frequency | integrated-time
\[\text{<Belief-Level> ::= certain disbelief | strong disbelief | moderate disbelief | weak disbelief | weak belief | moderate belief | strong belief | certain belief}\]

\[\text{<Evidence-Type> ::= accumulating | stand-alone}\]

\[\text{<Conclusion-Type> ::= crack flaw | volumetric flaw}\]

\[\text{<Remarks> ::= \{form-free text used to record the rule's author, comments, revision notes, etc.\}}\]

\[\text{<NIL> ::= \{no character; a null entry in the expansion\}}\]
3.6 Internal FLAP Rules

3.6.1 Introduction

The external rules described in the last section have a corresponding internal form which is more efficient to compute during automated flaw classification. In the FLAP implementation, efficient computation is defined in the context of a LISP-based inference engine. Therefore, the external rules are translated into valid LISP forms which can be decomposed using simple list commands and executed via the LISP "EVAL" statement. The rule number is preserved across the external/internal boundary for tracking purposes.

The FLAP inference engine evaluates the internal rules to accrue entries into one of four lists for accumulating and nonaccumulating evidence for the crack and volumetric hypotheses.

3.6.2 Syntax

Like the external rules, the internal rules are stored as text elements in a file which can be examined and/or modified by a text editor. This choice greatly facilitates validation of the translation process and allows discrete elements of the FLAP implementation to be tested separately from each other. The rules are stored as elements of a single list so that reads and writes of the rule set require just one LISP "READ" or "WRITE" command. Figure 3.7 lists the contents of an actual internal
\begin{verbatim}
((RULE 200 (SETQ NAC_CRK_EVD (CONS (* -0.7 (ZERO-CLIP (FEAT-CF (QUOTE POS_PULSE))) NAC_CRK_EVD)))

(RULE 202 (SETQ ACC_CRK_EVD (CONS (* 0.5 (ZERO-CLIP (FEAT-CF (QUOTE FLASH_PTS))) ACC_CRK_EVD)))

(RULE 204 (SETQ ACC_VOL_EVD (CONS (* 0.3 (ZERO-CLIP (FEAT-CF (QUOTE RINGING))) ACC_VOL_EVD)))

(RULE 206 (SETQ ACC_VOL_EVD (CONS (* 0.3 (ZERO-CLIP (FEAT-CF (QUOTE CREEP_WV))) ACC_VOL_EVD)))

(RULE 208 (SETQ ACC_CRK_EVD (CONS (* 0.3 (ZERO-CLIP (FEAT-CF (QUOTE RAYLEIGH_WV))) ACC_CRK_EVD)))

(RULE 402 (SETQ NAC_CRK_EVD (CONS (* 0.7 (ZERO-CLIP (FEAT-CF (QUOTE NORMAL_INCI))) NAC_CRK_EVD)))

(RULE 404 (SETQ ACC_CRK_EVD (CONS (* 0.5 (ZERO-CLIP (FEAT-CF (QUOTE DEEP_NULLS))) ACC_CRK_EVD)))

(RULE 406 (SETQ ACC_VOL_EVD (CONS (* 0.5 (ZERO-CLIP (FEAT-CF (QUOTE SHALLOW_NULLS))) ACC_VOL_EVD)))

(RULE 408 (SETQ ACC_VOL_EVD (CONS (* 0.5 (ZERO-CLIP (FEAT-CF (QUOTE SHARP_NULLS))) ACC_VOL_EVD)))
\end{verbatim}

Figure 3.7. FLAP Internal Rules
The grammar is specified using the same BNF syntax used in specifying the external rules.

<FLAP-Expr> ::= 
    (Rule <Rule-Number>
        (setq <Evidence-List-ID
            (cons (* <Rule-CF>
                (ZERO-CLIP <Feature-CF>))
            Evidence-List)))

<Rule-Number> ::= <Digit><Digit><Digit>

<Evidence-List_ID> ::= <Evidence-Type>_<Conclusion-Type>_EVD

<Evidence-Type> ::= ACC | NAC

<Conclusion-Type> ::= VOL | CRK

<Rule-CF> ::= <Sign><1-or-0><Digit>

<Sign> ::= - | + | <NIL>
39

<1-or-0> ::= 110

<Digit> ::= 0123456789

<Feature-CF> ::= (FEAT-CF '<Feature-ID>)
{Feat-CF is a function}

<Feature-ID> ::= POS_PULSE | FLASH_PTS
| RINGING | CREEP_WV
| RAYLEIGH_WV | NORMAL_INCI
| DEEP_NULLS | SHALLOW_NULLS
| SHARP_NULLS

3.7 FLAP Rule Translator

The FLAP rule translator is responsible for the mapping between the external rules and internal rules specified in the previous two sections. It is a program which is executed whenever the external rules are changed. The results of the translation, the internal rules, are stored in a file for repeated use by the inference engine. In this sense, the translator functions as a compiler rather than as an interpreter.
3.7.1 Design

Figures 3.8, 3.9 and 3.10 show the design of the translator program using simplified structure diagrams [5, 28]. Figure 3.9 expands on the highlighted function of Figure 3.8 and Figure 3.10 in turn expands on Figure 3.9. The simplicity of the design and implementation can be traced back to the constraints on the external form of the rules.

3.7.2 Implementation

The design is implemented in Symbolics Common Lisp as a set of 16 functions and 6 global data structures.

The highest level "Translate-Ext-To-Int" function is called with two arguments: (1) file name for the external rules and (2) file name for internal rules. The returned value is <NIL> for successful translation, Translation-Error otherwise.

The mapping between English phrases in the external rules and the corresponding tokens in the internal rules is done by a general-purpose recursive routine which makes use of Phrase-Value (P-V) data structures. These data structures are a list of ordered pairs where the first element of the pair is the English phrase and the second element is the corresponding LISP token.
Figure 3.8 FLAP Rules Translation: An Overview
Figure 3.9 FLAP Rules Translation: Translate All Rules
Figure 3.10 FLAP Rules Translation: Translate One Rule
A fragment of the P-V list used to map confidence phrases to numeric values looks like:

```
(Defconst Belief_CF_List
  '(((certain disbelief) -0.9)
  ((strong disbelief) -0.7)
  ...
  ))
```

A fragment of the P-V list used to map feature phrases to symbol names looks like:

```
(Defconst Feat_Name_List
  '(((a positive pulse) 'POS_PULSE)
  ((flash points are) 'FLASH_PTS)
  ...
  ))
```

If any of the mappings fail, i.e., the phrase being mapped is not found as the first element of any ordered pair in a P-V list, then the returned value is "*TR-ERROR*" which is inserted in the internal rule list. If this phrase is found by the validation function of the translator, the results are not stored in the file and an error is signaled.
3.8 FLAP Inference Engine

3.8.1 Introduction

The FLAP inference engine applies the rules to the facts (the features evaluated by FEAP) to arrive at a conclusion. If we consider the analogy of a courtroom, then FLAP consists of four elements: two attorneys, a judge, and a clerk. One attorney represents the rules seeking evidence that the flaw type is volumetric. The other attorney presents evidence that the flaw type is actually crack. The judge accumulates the evidence and pronounces a verdict based on the evidence presented. The clerk keeps a running account of the evidence and provides a history containing the rationale for the verdict.

The simplicity of the FLAP inference strategy can be traced to this courtroom analogy. The role of the attorneys is played by the rules themselves. And since the rules are written by the human domain experts, the inference strategy becomes derived from these domain experts rather than from the expertise of the programmer whose understanding of the problem domain may be less complete. The mechanism for using rules to present evidence is merely to examine each rule against the evidence uncovered by FEAP. The role of the judge is simply to accumulate the evidence presented and then decide on the conclusion favored by the most evidence. The weights used to consider each piece of evidence are again assigned by the human experts when they write the rules rather than by the programmer. The conclusion is presented as two confidence factors, one for each possible conclusion. The clerk watches the entire
process unobtrusively and makes notes which can be used to explain the verdict. These notes are then presented to the user as graphic displays on the terminal as a comprehensive audit trail.

3.8.2 Design

The design for the FLAP inference engine is given by Figures 3.11 through 3.14 where the shaded boxes in Figure 3.11 are expanded in the subsequent figures. Execution is driven by the number of rules and the number of look angles available. If the evidence for a particular rule is not found, then the rule has no effect. Likewise, if FEAP reports evidence that does not correspond to a FLAP rules, the evidence has no effect. There is no lower or upper limit on the number of look angles accommodated. When the number of look angles is zero, the confidence in both conclusions is zero. Figure 3.15 illustrates the control strategy.

3.8.3 Implementation

This design along with the explanation facility is implemented in Common Lisp as a set of 21 functions and 12 global data structures.

The highest level "FLAP" function is called with two arguments: (1) the pathname of the file containing the internal form of the FLAP rules and (2) the pathname of the file containing the output of FEAP. This function returns an error message if either file
FLAP Inference Engine

$n = \# \text{ of rules}$

$m = \# \text{ view angles}$

Figure 3.11 FLAP Inference Engine: An Overview
Figure 3.12 FLAP Inference Engine: Initialization
Figure 3.13 FLAP Inference Engine: Apply Rules to Facts
Figure 3.14 FLAP Inference Engine: Classify Flaw Type
Figure 3.15 FLAP Control Strategy
cannot be opened. Otherwise, FLAP displays its results on the terminal screen and creates the audit trail stored as a global list.

3.8.4 Run Time Environment

Two of the functions make up a run-time environment for evaluating the rules. These are "Zero-Clip" and "FEAT-CF" which are referenced from the internal form of the rules described in the previous section. The Zero-Clip function takes any confidence factor as its only argument and returns the maximum of that argument and zero. The significance of this is that FLAP considers affirmative FEAP evidence only. Negative evidence, i.e., disbelief that a particular feature exists in the flaw response, is ignored. This decision results from the observation made earlier about the differences between confidence factors and probabilities. Note however that this restriction on considering only affirmative evidence from FEAP does not extend to evidence "developed" by FLAP. For example, a positive belief that the leading edge polarity is positive causes the insertion of negative evidence into the FLAP nonaccumulating list supporting a crack conclusion.

FEAT-CF performs the matching between rules and evidence without any assumptions about the order of FEAP results. Because the feature set is so small (a result of choosing complex rather than simple features), the matching process is merely a recursive search on the FEAP output list.
3.8.5 Combining Evidence

The evidence within the nonaccumulating evidence lists is "summed" by simply picking the entry with the largest magnitude and using it to represent the non-accumulating evidence for a particular solution. This magnitude may be negative or positive.

The evidence within the accumulating evidence lists is "summed" using the algorithm developed for the MYCIN implementation [23]. When \( E \) represents the current confidence level, \( N \) represents the next piece of evidence expressed as a confidence factor, and \( E' \) represents the result of combining \( N \) with \( E \), then \( E' \) is given by:

\[
E' = E + (1-E) \times N
\]

Thus \( E' \) approaches 1 in asymptotic fashion when all the evidence is affirmative. Given a list of confidence factors to sum in this way, the final \( E' \) does not depend on the order of the list.

Conclusions are actually made at three different junctions. First, each look angle is examined, in isolation from other look angles, to reach the best possible confidence in the two propositions: (1) these look angle data are from a crack-like flaw and (2) these look angle data are the result of a volumetric flaw. Secondly, these two hypotheses are resolved into one single conclusion for this look angle data. A propagated value is
calculated at this junction and accumulated into a list of volumetric and crack conclusions. Finally, these lists are given a weighted sum and then compared to each other to make a conclusion about the flaw type based on all the evidence examined so far.

For the first junction, the nonaccumulating and accumulating evidence lists for each conclusion are "MYCIN-Summed" as discussed above. Then, the two sums for each conclusion are "Magnitude-Summed" by choosing the largest magnitude. The proposition with the highest resulting confidence factor is the winner.

For the second junction, the two propositions are compared to each other and a single proposition is chosen to the conclusion for the look angle. This "winning" conclusion is forwarded to the next junction by putting the Propogated-CF of the winning proposition into one of two lists that exist to accumulate crack and volumetric conclusions. The Propogaed-CF value is obtained from the "win/lose" formula:

\[ \text{Propogated-CF} = [\text{Winner-CF} - \text{Loser-CF}] \]

Thus, if the belief levels in the two conclusions are nearly equal, the net effect is to propagate nothing. This corresponds to the notion that data from look angles which are inconclusive should not affect the final conclusion.

Finally, these propogated conclusions are summed in a "Weighted-MYCIN-Summation", and then combined with the "win/lose" formula to arrive a single consensus conclusion that reflects all the data seen so far.
Figure 3.16 shows this "summation" process where "MYC +" represents "MYCIN-Summing", "MAG +" represents choosing the largest magnitude, "Win/Lose" refers to the algorithm of that name, and "WT MYC +" represents the "Weighted-MYCIN-Summation" process.
Figure 3.16. FLAP Evidence "Summation"
Figure 3.16 Continued.
Lists of Conclusions for Look Angles 1..i
(from the previous page)

Crack Conclusions

Volumetric Conclusions

WT MYC

WT MYC

Win/Lose

Crack Conclusion

Volumetric Conclusion

Single Consensus Conclusion for Look Angles 1..i

Figure 3.16 Continued.
3.9 Flaw Type Results

FLAP execution results fall into two categories: (1) results meant for real-time display, i.e., while FLAP is executing and (2) results used to validate FLAP conclusions. This section discusses the nature of both types in FLAP. If the results are themselves viewed as knowledge, then FLAP representation of this knowledge is bi-level, i.e., one level (real-time) is meant for use by the NDE inspector evaluating a particular flaw while the other level is meant for use by the NDE engineer responsible for creating the FLAP rules used in the evaluation. The display of the results and format of the outputs themselves are discussed in the following subsections.

3.9.1 Real Time Results

As FLAP evaluates each look angle data set, it accumulates and displays the results of that evaluation in terms of four elements: Confidence in the hypothesis that the flaw response from this look angle is the result of a crack type flaw, confidence in the hypothesis that this flaw response is the result of a volumetric flaw, a combined conclusion for this look angle, and a consensus conclusion for all the look angles examined so far. The consensus conclusion is calculated using the "win/lose" algorithm described in the previous section.
3.9.2 Example

An example will clarify this discussion. Suppose the following parameters exist just after evaluation of all the rules against the features found in the first look angle of the data set.

Crack-Conclusion confidence = +0.50
Volumetric-Conclusion confidence = +0.20
Cumulative-Conclusion = unknown with confidence 0.0

Then the conclusion for look angle 1 is Crack with confidence +0.30 (0.50 - 0.20) and Cumulative-Conclusion is Crack with confidence +0.30. Now, consider the impact of evaluating the features in the second look angle.

Crack-Conclusion confidence = +0.10
Volumetric-Conclusion confidence = +0.30
Cumulative-Conclusion = Crack with confidence +0.30.

Then the conclusion for look angle 2 is Volumetric with confidence +0.20 (0.30 - 0.10) and Cumulative-Conclusion is Crack with confidence +0.10 (0.30 - 0.20). Consider the results for the third look angle.

Crack-Conclusion confidence = +0.30
Volumetric-Conclusion confidence = +0.80
Cumulative-Conclusion = Crack with confidence +0.10.

The conclusion for look angle 3 is Volumetric with confidence +0.50 and Cumulative-Conclusion is Volumetric with confidence +0.25 ([0.10 + (1 - 0.10)*0.50] - 0.30).

3.9.3 Validation Results

To satisfy the requirements for category (2) results, FLAP keeps track of which rules fire, how often they fire, and how much they contribute to the final conclusion. The Explanation Facility uses this information to display a summary of rule usage to the operator after the last look angle has been examined. This information is also used to construct the audit trail.

3.10 FLAP Explanation Facility

The Explanation Facility seeks to explain how FLAP reached its conclusions in terms of the expert rules. Thus, the Explanation Facility makes FLAP a visible "white box" as contrasted to the "black box" appearance of adaptive learning systems.

Initially, the Explanation Facility was envisaged as a post-processing module that would be invoked only after the last look angle had been evaluated. In this scenario, the screen displays would be summary information derived from the more complete audit trail. After consultation with the Center's industrial sponsors, the design was
changed to display the four real-time results described in the previous section during execution of FLAP. The sponsor's argument was that in an industrial setting, the number of look angles required to inspect a particular structure must be minimized. Thus, they wanted information from FLAP on what that minimum number was.

The number of look angles required to characterize a particular flaw type is dependent on the structure geometry, the test equipment, and flaw itself. So, the design was modified to display the cumulative conclusion after evaluating each angle with the intention that industrial sponsors could then set a policy of taking $X+Y$ measurements from different look angles where $X$ is some absolute minimum, at least 1, and $Y$ is the number of measurements needed to reach some threshold confidence level. For example, company A might decide to always take two measurements and then continue to take measurements at varying look angles until the confidence level for crack or volumetric exceeds +0.60. To meet these requirements, the Explanation Facility was incorporated into the FLAP Inference Engine.

Some explanation facilities are designed to be interactive. They have command sets that may include "Why?" and "How?". This design is not interactive because we anticipate the need to answer "Why?" long after the inspection run is complete such as in the case of in-service structural failures. This need for postmortem "Why?" analysis is met by the audit trail which may be saved on disk or as hardcopy by the inspector.
3.11 Summary of Results

The display of the real-time results on the operator terminal serve two purposes. First, the display is consistent with the goal described in the previous section of informing the inspector when enough look angle measurements have been collected. Second, the display gives the inspector an indication of how good the flaw type characterization is in terms of converging conclusions and the number of rules supporting the final conclusion. Figure 3.17 shows the four display windows which appear on the operator's terminal.

3.11.1 Conclusions Window

This display contains the four real-time FLAP results described previously. The display format serves the first purpose above if one thinks of two lines drawn at the desired confidence level for the volumetric and crack conclusions. When the connected hollow square points cross either line, then enough look angles have been collected.

Uncertainty in the final conclusions are signaled by symmetry of the solid dots about the zero-axis. This uncertainty also causes the distance between the hollow triangles and the solid dots to increase. If the conclusion is sensitive to the chosen look angle, then the scattering of the plotted triangles will increase.
Figure 3.17. FLAP Display Window
3.11.2 Firing Percentage

This display is a simple check on the quality of the rules as they apply to this evaluation. When many rules fire, i.e., have an impact on the conclusions, then a large part of the knowledge base took part in the inference. When only a few rules fire, then only a small part of the knowledge base was actually utilized. In this case, the conclusion must be examined more critically.

3.11.3 Source of Accumulating Crack/Volumetric Evidence

These two displays indicate the source of the accumulating evidence used to support each hypothesis. This display tells the operator as well as the developer which rules had the most leverage for these data. The values in this display depend on the frequency of firing as well as the weight assigned to the rules.

3.12 Audit Trail

The audit trail display is simply a listing of the FLAP Audit_Trail data structure that is built during execution. The audit trail serves two purposes. First, it can be used by a developer to fine-tune the system in terms of rule weights. Recall that this system was designed to be a starting point for industrial users to develop their own customized expert systems. Second, the audit trail for a particular inspection which has been printed and filed can be used by investigators trying to assess the reliability of the
system in the light of structural failures. This type of accountability is crucial to acceptance of heavily regulated industries such as those who build and operate nuclear power plants.

Figure 3.18 shows a sample audit trail. The header describes the execution environment in terms of the software version number, actual pathnames to the files containing the rules and feature evaluations, and the date time group. The contribution of each rule is described in terms of type of evidence, conclusion supported, and the numerical value of the contribution. After each rule is evaluated against the features for a particular view angle, the resulting conclusions are recorded.
This is the audit trail of a FLAP [Version 3.1] run at 01/18/89 15:18:25.

The data set filename is Internal HD80:flex:deliverables: FLAP:TestData:feap_feat_eval.t44.

The rule set filename is Internal HD80:flex:deliverables: FLAP:Test Data:intrules2.text.

Rule 200 [POS_PULSE] fired. The resulting contribution to NAC_CRK_EVD is -0.35.

Rule 202 [FLASH_PTS] fired. The resulting contribution to ACC_CRK_EVD is 0.25.

Rule 204 [RINGING] missed. The resulting contribution to ACC_VOL_EVD is nothing.

Rule 206 [CREEP_WV] fired. The resulting contribution to ACC_VOL_EVD is 0.09.

Rule 208 [RAYLEIGH_WV] fired. The resulting contribution to ACC_CRK_EVD is 0.15.

Rule 402 [NORMAL_INCI] missed. The resulting contribution to NAC_CRK_EVD is nothing.

Rule 404 [DEEP_NULLS] fired. The resulting contribution to ACC_CRK_EVD is 0.15.

Rule 406 [SHALLOW_NULLS] fired. The resulting contribution to ACC_VOL_EVD is 0.15.

Rule 408 [SHARP_NULLS] missed. The resulting contribution to ACC_VOL_EVD is nothing.

Angle 1 findings:
Confidence in crack is 0.46.
Confidence in volumetric is 0.23.

Angle 1 conclusion:
Flaw type is crack with confidence 0.23.

Angles 1 through 1 conclusion:
Flaw type is crack with confidence 0.23.

Figure 3.18. FLAP Audit Trail
Rule 200 [POS_PULSE] fired.
The resulting contribution to NAC_CRK_EVD is -0.21.

The resulting contribution to ACC_CRK_EVD is nothing.

Rule 204 [RINGING] fired.
The resulting contribution to ACC_VOL_EVD is 0.15.

Rule 206 [CREEP_WV] fired.
The resulting contribution to ACC_VOL_EVD is 0.09.

Rule 208 [RAYLEIGH_WV] fired.
The resulting contribution to ACC_CRK_EVD is 0.09.

Rule 402 [NORMAL_INCI] missed.
The resulting contribution to NAC_CRK_EVD is nothing.

Rule 404 [DEEP_NULLS] fired.
The resulting contribution to ACC_CRK_EVD is 0.15.

Rule 406 [SHALLOW_NULLS] fired.
The resulting contribution to ACC_VOL_EVD is 0.15.

Rule 408 [SHARP_NULLS] missed.
The resulting contribution to ACC_VOL_EVD is nothing.

Angle 2 findings:
Confidence in volumetric is 0.34.
Confidence in crack is 0.23.

Angle 2 conclusion:
Flaw type is volumetric with confidence 0.12.

Angles 1 through 2 conclusion:
Flaw type is crack with confidence 0.06.

Figure 3.18 Continued.
Angle 3 findings:
  Confidence in volumetric is 0.62.
  Confidence in crack is -0.35.

Angle 3 conclusion:
  Flaw type is volumetric with confidence 0.62.

Angles 1 through 3 conclusion:
  Flaw type is volumetric with confidence 0.37.

Angle 12 findings:
  Confidence in crack is 0.35.
  Confidence in volumetric is 0.32.

Angle 12 conclusion:
  Flaw type is crack with confidence 0.03.

Angles 1 through 12 conclusion:
  Flaw type is volumetric with confidence 0.79.

Rule 200 [POS_PULSE] fired.
  The resulting contribution to NAC_CRK_EVD is -0.21.

  The resulting contribution to ACC_CRK_EVD is 0.25.

Rule 204 [RINGING] fired.
  The resulting contribution to ACC_VOL_EVD is 0.15.

Rule 206 [CREEP_WV] fired.
  The resulting contribution to ACC_VOL_EVD is 0.21.

Figure 3.18 Continued
Rule 208 [RAYLEIGH_WV] fired.
The resulting contribution to ACC_CRK_EVD is 0.09.

Rule 402 [NORMAL_INCI] missed.
The resulting contribution to NAC_CRK_EVD is nothing.

Rule 404 [DEEP_NULLS] missed.
The resulting contribution to ACC_CRK_EVD is nothing.

Rule 406 [SHALLOW_NULLS] fired.
The resulting contribution to ACC_VOL_EVD is 0.35.

Rule 408 [SHARP_NULLS] missed.
The resulting contribution to ACC_VOL_EVD is nothing.

Angle 13 findings:
- Confidence in volumetric is 0.56.
- Confidence in crack is 0.32.

Angle 13 conclusion:
- Flaw type is volumetric with confidence 0.25.

Angles 1 through 13 conclusion:
- Flaw type is volumetric with confidence 0.80.

Figure 3.18 Continued.
4 RESULTS

This section describes the experimental results obtained by executing FLAP.

4.1 Translator

Verification of the Translator software was accomplished by inspecting the internal rules generated for a given set of external rules. Deliberate errors in the external rules were detected while error-free external rules translated correctly into executable internal rules.

4.2 Inference Engine

4.2.1 Introduction

Results are shown to demonstrate correctness of the implementation. The correctness of the rules is not the main concern of this computer engineering thesis. Correctness of the calculations was validated by comparing each entry in the audit trail of a FLAP run with the results which were pre-calculated by hand. This micro-level verification was successful. Higher-level verifications were performed for two volumetric flaw and three crack flaw data sets. Two representative test cases are T44 for volumetric data and C02 for crack data.
4.2.2 Volumetric Flaw (T44)

This actual flaw is a 200 X 400 micron oblate spheroid in titanium. Thirteen look angles are available.

Figure 4.1 shows the FLAP output screen for features evaluated by human experts during the development of FLAP. The data of look angle 1 were judged to be from a crack. The data of 3 other look angles were inconclusive. The remainder of the look angles were evaluated as characteristic of a volumetric flaw. The final conclusion for all the look angles was volumetric with confidence +0.80.

Figure 4.2 shows the results for the same data set except that the ordering of the look angles is reversed, i.e., the data for look angle 1 are now look angle 13 and so forth. Note that the shape of the top curve is changed, but the final conclusion is independent of look angle order. The bar graph outputs are also independent of the look angle order.

Figure 4.3 shows the results for the original T44 data set except that in this case, the feature evaluations are from FEAP. The overall conclusion is essentially unchanged but the pattern of which rules were used the most, exhibited by the bar graphs, is significantly different.
Figure 4.1. FLAP Screen Results for Manual Feature Evaluation of T44 Data Set
Figure 4.2. FLAP Screen Results for Manual Feature Evaluation of T44R Data Set
Figure 4.3. FLAP Screen Results for FEAP Feature Evaluation of T44 Data Set
4.2.3 Crack Flaw (C02)

This actual flaw is a 40 micron radius disk-shaped cavity in titanium. Sixteen look angles are available.

Figure 4.4 shows the results for this data set using human experts to evaluate the features. FLAP correctly inferred the flaw type on all of them. The final conclusion for all look angles was crack with confidence +0.92.

Figure 4.5 shows the results for the same data when FEAP was used to evaluate the features. Again, the final conclusion is nearly the same even though the rule distribution is changed.

Figure 4.6 shows the results for this data set when the look angle data for look angles 6 through 16 are made zero. In this case, the confidence decays towards zero demonstrating that confidence is active rather than passive. This corresponds to the human intuition that uncertain results should lower the confidence in our conclusions.
Figure 4.4. FLAP Screen Results for Manual Feature Evaluation of C02 Data Set
Figure 4.5. FLAP Screen Results for FEAP Feature Evaluation of C02 Data Set
Figure 4.6. FLAP Screen Results for Manual Feature Evaluation of C02Z Data Set
5 SUMMARY AND DISCUSSION

This section provides a brief summary of the research and discusses three avenues for future research.

5.1 Summary

The goal of this research was to demonstrate the feasibility of applying AI techniques to a specific problem in the domain of nondestructive evaluation. This goal was completely satisfied by the design and implementation of FLEX. The FLAP portion of FLEX demonstrates, in particular, a strategy for using rule-based expert systems to solve open problems in industry.

As of this writing, no other system with FLAP's capabilities is known to this author. Written comments by the Center's industrial sponsors indicate a high degree of interest and satisfaction with the results obtained. The results of this research have been reported in technical meetings with the sponsors, in international conferences, in the conference proceedings, and in a technical journal.

5.2 Discussion

This section discusses modifying FLAP to include a learning component, possibilities for parallel execution of FLAP, and a simple equivalent neural network model of FLAP.
5.2.1 Learning with Feedback

Two kinds of learning are applicable to FLAP. First, FLAP could be modified to learn which features are associated with each flaw type. This type of learning is not necessary for the flaw type determination problem however since the theoretical basis for associating certain features with a given flaw type is well understood. A second FLAP requirement is learning how strongly a given feature is associated with a given flaw type. This learning is reflected in adjusting, or tuning, the value of the confidence factors associated with each rule.

This tuning process could be automated by employing the feedback scheme shown in Figure 5.1. In this scheme, the flaw response for a known flaw is evaluated by FEAP in normal fashion and then examined against the internal form of FLAP rules by the FLAP inference engine. The results are compared to the correct answer or goal established by the human expert. A new function, the FLAP Learner, compares the inference results to the correct answer. If FLAP reached the wrong conclusion for the entire set of view angles, then rule confidence factors are adjusted until the correct answer is reached. If FLAP is already reaching the correct conclusion (the most likely case), then the confidence factors in the rules are adjusted to increase FLAP's confidence in its correct conclusion.

The learner chooses confidence factors to modify based on agreement with the correct answer and by looking at which rules have the most leverage for the test data
Figure 5.1. A FLAP Learning Model Employing Feedback
being examined. Only the internal form of the rules is modified in this iterative process of test, compare, and modify. The set of feature evaluations is fixed during this process.

When the learner finds a better set of confidence factors for the rules, it causes the internal form of the rule to be translated back to external form for examination by the human expert. This backwards translation is possible by using the rule number from the internal rule as a key to retrieve the English phrasing from the corresponding external rule. If the human expert agrees with the learned rules, then he or she can incorporate them into the configured external rules.

5.2.2 Parallel Execution

When automated inspection systems have to keep pace with moving assembly lines, expert systems such as FLEX must meet strict throughput requirements. In this discussion, I mention two architectural modifications which could be used to shorten execution time.

Figure 5.2 shows a method of using multiple processing units to evaluate each FLAP rule in parallel with every other rule. This parallel execution is made possible by the fact that FLAP rules are all independent from each other. Additional processing units would be required to dispatch the feature evaluations to the appropriate rule processors and to collect and explain the results from the blackboard [1] memory which in turn is fed by each of the rule processors. In this architecture, the task assigned to
Figure 5.2. Parallel Processor Execution of FLAP
each CPU is so small that the individual processor requirements could be met with existing 16-bit microprocessors.

While this architecture has the highest (reasonable) degree of parallelism for FLAP, it is not the best solution. Specialized hardware architectures, like the one in Figure 5.2, are expensive to develop and maintain. Dedicating one processor to one rule is overkill when the rule evaluations are so simple. A better system approach would be to break up the FLEX problem according to each feature and then combine related elements of FEAP and FLAP into independent tasks which can be executed in parallel. Unfortunately, this is not possible since in the FEAP decision trees, the results of evaluating the flaw response for the feature FLASH_PTS are used to evaluate the feature RINGING. That is, FEAP evaluations are not independent of each other.

A more feasible approach to speeding up FLEX is to use pipelining as shown in Figure 5.3. In this approach, four processing systems are used to execute, in parallel, the necessary programs for data acquisition, FEAP, FLAP inference, and FLAP explanation facility. The four processing systems could either be networked personal computers or a four-element processor like the Apollo DN1040. The computational load on the fourth processor is light enough that it could be used to control the network or as the operating system server in a multiprocessing environment. No changes to the software would be required. This parallelism is possible because FEAP, FLAP inference, and FLAP explanation tasks are all independent from one another.
Figure 5.3. FLEX Pipelined Execution
5.2.3 Using a Neural Net Instead of an Expert System

Although I will not show it in this thesis, I suspect that the functionality of any classical expert system can be modeled in a neural net, but the reverse transformation is not always possible. Figure 5.4 shows how the functionality of FLAP could be represented in a simple neural net. Solid lines from rule nodes indicate activation or excitation. Dashed lines indicate suppression. The output of most rules is "summed" by a special accumulating transfer function. Rules 200 and 402 are necessary and sufficient respectively and therefore link directly to the crack conclusion. The drawback of this approach is that the weighting of each rule is now represented (less explicitly) in the links between nodes rather than in the text of an English-like rule in the expert system.

Figure 5.5 shows a fragment of a neural net which cannot be translated to FLAP-style rules. Linkages between peer nodes of a neural net, i.e., cycles, are permissible so long as the net eventually stabilizes due to convergence or a damping transfer function. The linkage can be captured in the rules shown in Figure 5.5, but these rules cannot be evaluated just once in any order. Going back to the courtroom analogy, rules like this would require several cycles of examine, rebut, examine, rebut, and so forth.

The author plans to investigate neural nets, particularly the problem of how to make their knowledge more explicit to the human expert, in the Ph.D. program of research.
Figure 5.4. Expressing FLAP as a Simple Neural Net
Rule 600: if "Y" then
  "X" and
  "Z"

Rule 602: if "Z" then
  "X" and
  "Y"

Figure 5.5. Expressing a Neural Net Fragment as Rules
6 BIBLIOGRAPHY


7 ACKNOWLEDGEMENTS

This research was made possible by the many people who have given me instruction, advice, guidance, and friendship. I am deeply indebted to all of them. I want to especially acknowledge the contributions of my colleagues Ken Christensen, Lat Koo, and Brian Lovewell; and of my research advisor at the Center for NDE, Dr. Lester Schmerr.

I am also deeply grateful to Dr. Charles Wright for his guidance and support and to Dr. Terry Smay for his advice at crucial junctures. I thank Dr. Deepinder Sidhu for his early encouragement of my efforts.

Finally, I want to express my heartfelt appreciation and admiration to my wife Debbie for her steadfast support and unwavering commitment to see my efforts through to their conclusion.

This research was supported by the Center for Nondestructive Evaluation at Iowa State University and was performed at Ames Laboratory which is operated for the U.S. Department of Energy by Iowa State University under Contract No. W-7405-ENG-82.