Self-adaptation and Rule Generation in a Fuzzy System for X-ray Rocking Curve Analysis

By

Tony Partridge

Submitted for the degree of Doctor of Philosophy

to the Higher Degrees Committee

University of Warwick

Department of Engineering

University of Warwick, U. K.

October 1994

CONTENTS

List of figures

List of tables

Acknowledgements

I would like to thank my supervisor Dr. T. Tjahjadi for all his comments, suggestions and advice, but most importantly for always being available to discuss my work. At critical moments, Dr. Tjahjadi's comments have helped to crystallise important ideas in my mind. I am particularly grateful for this.

I would like to thank the domain experts Prof. D. K. Bowen, Prof. B. K. Tanner, Dr. N. Loxley and Dr. C. R. Thomas for their help and contributions. I would also like to express my thanks to Bede Scientific Instruments Ltd., U.K., for partial funding of this research.

I would like to thank Mr. P. Cooney, Dr. R. Dale-Jones and Mr. M. Crean.

I would like to thank Dr. F. Cooney for persuading me, advising me and sustaining me during the lows and supporting me during the highs of the past four years.

Declaration

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy by the Higher Degree Committee at the University of Warwick. The thesis has been composed and written by myself based on the research undertaken by myself. The research materials have not been submitted in any previous application for a higher degree. All sources of information are specifically acknowledged in the content.

T. Partridge

Summary

X-ray rocking curve analysis is an example of a changing application domain. The salient characteristic of such a domain is that situations and facts can change over time. This means that the domain cannot be modelled by a fixed set of fuzzy rules. Instead, the rules must change over time and these changes must model actual changes that occur in the application domain. Three new techniques have been developed for altering a set of fuzzy rules: altering the credibility weight of an expert and using connection matrices to shift the focus of attention between different sets of rules; fine-tuning and changing the membership functions of fuzzy premise variables and thereby altering the meaning of the rules; and generating new fuzzy rules by inductive learning from examples.

A fuzzy system for X-ray rocking curve analysis has been developed and used to test each of these techniques. This fuzzy system uses frames, logic-based variables, connection matrices and credibility weights, fuzzy rules and a record of previous decisions in order to model X-ray rocking curve analysis. Question and answer sessions with the user are used to describe experimental rocking curves and structural parameters are deduced from this description. These structural parameters are then used to simulate a theoretical curve, which is compared with the experimental one. A performance measure is derived to calculate the degree of matching between the two curves. This performance measure is used to test each of the three techniques in turn. Tests have shown that the fuzzy system optimises its performance to suit new situations and facts.

CHAPTER 1. Introduction.

1.1 Introduction.

This thesis introduces three new techniques that allow a fuzzy system to make decisions in a changing application domain. A fuzzy system is an intelligent system that uses fuzzy if/then rules to model imprecise and uncertain concepts (Zadeh 1992b). Deductions are made about these concepts and these deductions are used to solve problems in an application domain. For example, fuzzy systems have been used to control a plant (Mamdani 1975, Berenji 1992), to make a diagnosis of a patient's symptoms (Sanchez 1992) and to interpret and model uncertainty in computer vision (Keller and Krishnapuram 1992). In each of these cases, the fuzzy systems were applied in static application domains.

A static application domain is one that does not change over time. In these types of domains, the relationships between inputs and outputs to a fuzzy system can be modelled using a fixed set of fuzzy rules. A changing application domain is one where the relationships between inputs and outputs to a fuzzy system can change over time. Deductions that are correct for the current inputs and outputs can be incorrect for later ones. A fixed set of fuzzy rules cannot model this type of behaviour. Instead, the fuzzy system needs to adapt and change its rules in order to follow the changes that occur in the application domain.

Fuzzy rules use vague concepts in the premises and a single vague concept in the consequences of the rules. Input data is presented to the fuzzy system and a set of outputs is inferred. Fuzzy rules are derived using two methods: elicitation of heuristics from human experts; and directly from data using statistical means (Kosko 1992a). In a static application domain, once the rules are derived they remain fixed and do not change over time. In a changing application domain, the rules must change.

Three methods can be identified for adapting and changing rules in a fuzzy system. The first method is to shift the focus of attention between different sets of rules. The second is to alter the meaning of individual rules. The third method is to generate new fuzzy rules. In this thesis, three new techniques are introduced for implementing each of these methods. In the first technique, a set of experts' credibility weights are evaluated and incrementally altered to suit the most recent decisions made by the fuzzy system. These credibility weights are used to choose the most appropriate fuzzy rules to use in a decision. In the second technique, individual fuzzy rules are fine-tuned and changed in order to optimise the performance of the rules. In the third technique, inductive learning from examples is used to generate new fuzzy rules from an example set of recent inputs and outputs taken from good decisions made by the fuzzy system. These techniques are demonstrated using a fuzzy system for X-ray rocking curve analysis. This is an example of a changing application domain.

An X-ray rocking curve is the spectrum obtained when a single crystal is rotated in an X-ray beam through an angle at which it diffracts strongly (Halliwell, Lyons and Hill 1987). It is highly sensitive to the details of the structure of the outer layers of the material. The measurement and analysis of double X-ray rocking curves are widely used in research and in industrial production as a means for investigating the perfection of a wide variety of natural and synthetic crystals (Bowen, Hill and Tanner 1987). The number of different structures that can be analysed, and the different relationships between experimental rocking curves and their underlying structures, is computationally infinite. It is impossible to prescribe for all these possibilities in a fixed set of rules.

A fuzzy system for X-ray rocking curve analysis has been developed. This fuzzy system uses the three new techniques to adapt a set of fuzzy rules to a changing application domain. The purpose of this thesis has been to introduce these three new techniques, to demonstrate their use in a particular changing application domain and to test each of the techniques using sample data taken from the domain.

1.2 Organisation of the thesis.

Chapter 2 is a review of the techniques of the calculus of fuzzy if/then rules as applied in fuzzy systems. The techniques of fuzzification and defuzzification are reviewed and then the review is extended to deal with the situation where fuzzy systems are applied in changing application domains. Three methods for adapting and changing fuzzy rules are identified and existing techniques are reviewed that attempt to implement these methods. In each case, arguments are given to justify the development of new techniques.

Chapter 3 is a review of X-ray rocking curve analysis, an example of a changing application domain, taken from the literature. The review includes a description of the problem, the usual method of solution, the problems associated with this method and a justification for the development of a fuzzy system for this application.

Chapter 4 presents a new technique for changing the focus of attention in a fuzzy system. This technique incrementally alters experts' credibility weights on the basis of recent decisions involving the rules of these experts. A mathematical equation is defined to calculate the size of changes and the technique is tested using a fuzzy system for X-ray rocking curve analysis. A particular limitation of this technique is

discussed and one solution to the problem is presented at the end of the chapter.

In Chapter 5, two techniques for fine-tuning and changing fuzzy rules are presented. The first technique uses a simplified version of fuzzy rules. These rules use discrete values in the premises of the rules. However, because this technique restricts the deductive power of the underlying fuzzy logic, a second technique is presented. The latter technique uses vague linguistic labels in the premises. Both techniques are tested using a fuzzy system for X-ray rocking curve analysis.

Chapter 6 presents a new algorithm for generating new fuzzy rules using inductive learning from examples. Six fitness functions based on well-established evaluation criteria are derived and used to test new rules. This algorithm is tested using examples taken from a fuzzy system for X-ray rocking curve analysis.

A fuzzy system for X-ray rocking curve analysis is presented in Chapter 7. Object-oriented programming and a number of knowledge representation techniques are used to implement the fuzzy system. A performance measure for X-ray rocking curve analysis is also derived. The fuzzy system is tested using 100 sample decisions.

Chapter 8 presents a number of benefits and limitations of the three new techniques. A number of areas for possible future work are also presented.

CHAPTER 2. Review of fuzzy systems in changing application domains.

2.1 Introduction.

Fuzzy logic deals with imprecise, uncertain and unreliable knowledge. It relates vague linguistic descriptions of a concept to a partial membership that takes values ranging from 0 to 1. These values are the degree to which a particular input matches a vague concept. A strict mathematical framework has been derived to model this (Zadeh 1992a, Yager 1992, Kosko 1992a), but for most practical applications a relatively restricted part of fuzzy logic called the calculus of fuzzy if/then rules is used (Zadeh 1992b). In this calculus, a set of fuzzy rules is created so that these rules estimate the relationships between inputs and outputs to the system. Deductions are made by firing rules concurrently. An intelligent system that uses this calculus is called a fuzzy system.

In the case where a fuzzy system is operating in a changing application domain, changing or time-related knowledge will be presented to the fuzzy system. These types of knowledge will alter the relationships between inputs and outputs to the fuzzy system. As new relationships occur, a fixed set of fuzzy rules will become ineffective. This is because the fixed set of rules describes the original relationships between inputs and outputs. These rules fail to describe the new relationships. Therefore, when application domains change, the fuzzy rules that model these domains will also need to change.

There are three ways in which a fuzzy rulebase can be changed. Firstly, the focus of attention can be shifted from one set of fuzzy rules to another. In this way, the performance of the fuzzy system is optimised by choosing those rules that have been successful in the recent past. The second way in which a fuzzy rulebase is changed is to fine-tune and alter the meaning of the fuzzy rules. This customises the

existing rules to suit the most recent relationships between inputs and outputs. The third way is to generate new fuzzy rules from an example set of inputs and outputs to the fuzzy system. Inductive learning from examples is the most effective method for generating new fuzzy rules in a changing application domain. This is because it focuses on the most recent examples and uses these to create relevant rules. But new rules must also be tested before they are incorporated into the fuzzy rulebase. Fitness functions based on evaluation criteria for intelligent systems are used to test new rules. These fitness functions are mathematical formalisations of these evaluation criteria. They give a rigorous and objective measure of the degree to which a new rule satisfies the evaluation criteria.

2.2 Fuzzy systems.

Fuzzy systems use the calculus of fuzzy if/then rules to represent knowledge and make deductions (Zadeh 1992b). This calculus is a fairly self-contained collection of concepts and methods for handling knowledge that is imprecise, uncertain or unreliable. The general form of a fuzzy rule is as follows:

$$\begin{aligned} &\text{IF} && X_1 = A_1 \text{ AND } X_2 = A_2 \ldots \text{ AND } X_n = A_n \\ &\text{THEN} && Y = B \end{aligned} \qquad (2.1)$$

where $X_1, \ldots, X_n$ are fuzzy premise variables, $A_1, \ldots, A_n$ are linguistic labels, Y is a fuzzy consequent variable and B is a linguistic label. The logical OR connective can be substituted for the AND connective in this rule. Figure 2.1 shows a simple example of a fuzzy rule for boiling an egg.

```
IF          temperature of the water = VERY HOT
      AND   time of cooking = VERY LONG
THEN  egg = HARD
```

Figure 2.1 Example of a fuzzy rule for boiling an egg.

A fuzzy rule is defined in terms of vague linguistic labels, such as VERY HOT and VERY LONG. Fuzzy rules are fired concurrently and these deductions are then used to solve particular problems in an application domain (Kosko 1992a). What is important in practical applications is that the fuzziness of the antecedents eliminates the need for a precise match with the input. As a consequence, in a fuzzy system, each rule is fired to a degree that is a function of the degree of match between its antecedent and the input. Fuzzy systems have been used in a number of application areas, e.g. controlling a plant (Mamdani 1975, Berenji 1992), diagnosing of a patient's symptoms (Sanchez 1992) and interpreting and modelling of uncertainty in computer vision (Keller and Krishnapuram 1992).

2.2.1 Fuzzification.

The inputs to a fuzzy system are sets of discrete values. These discrete values are passed to fuzzy variables in the premises of fuzzy rules. Each fuzzy variable has a set of membership functions which corresponds to a set of linguistic labels. In practice fuzzy engineers have found triangular and trapezoidal shapes help capture the modeler's sense of fuzzy numbers and simplify computation (Kosko 1992a). But monotonic and bell-shaped functions have also been used (Berenji 1992). Each time a value is passed to a fuzzy premise variable, that value is fuzzified. This means the value is tested against the membership functions of the fuzzy variable and the level

of truth for each triangular function is calculated. There is one triangular function associated with each linguistic label and the calculated value, or the level of truth, for this triangular function is considered the support for this linguistic variable. The set of linguistic labels for a fuzzy variable is called its term set (Zimmermann 1991).

For a fuzzy premise variable A, a fuzzy membership function $m_A : Z \rightarrow [0, 1]$ assigns a real number between 0 and 1 to every element z in the universe of discourse Z. This number $m_A(z)$ indicates the degree to which the data z belongs to the fuzzy set A. $m_A(z)$ is also called the test score for the variable A (Zadeh 1987). As an example, assume the membership functions $m(z)$ for the fuzzy premise variable A as shown in Figure 2.2.

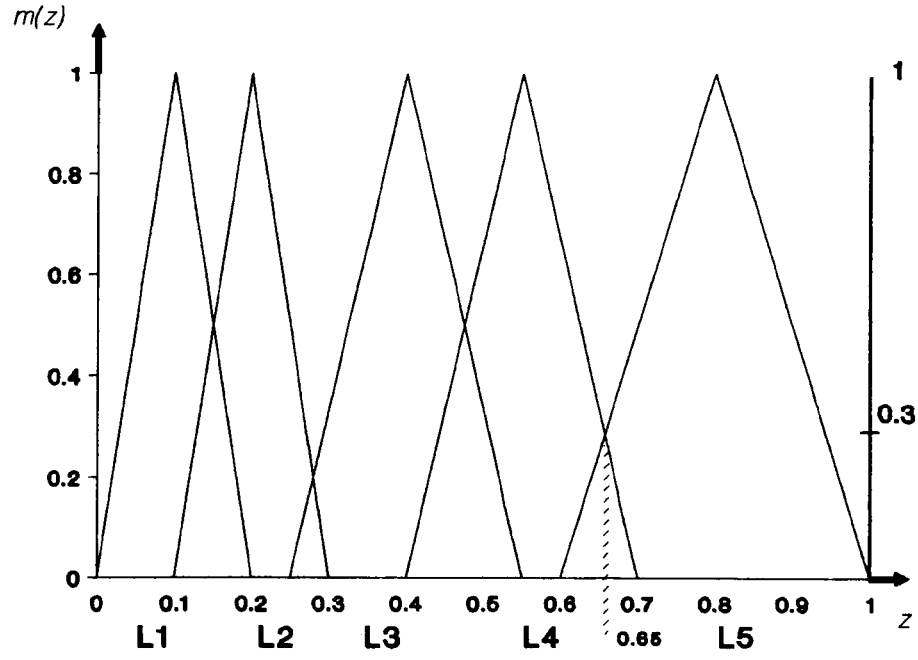

Figure 2.2. Triangular membership functions for a fuzzy premise variable A. Each triangle is associated with a linguistic label L1, L2, L3, L4, L5.

When the input value for this variable is 0.65, this produces the following levels of

membership: {L1 = 0.0, L2 = 0.0, L3 = 0.0, L4 = 0.3, L5 = 0.3}. {L1, L2, L3, L4, L5} is the term set for A. For each fuzzy variable A, the fuzzy measure $m_A(z)$ is defined. This measure is the degree to which the variable is compatible with its intended meaning. For example, the fuzzy variable *temperature_of_the_water* with the term set {EXTREMELY HOT, VERY HOT, FAIRLY HOT, TEPID, COLD}, will be a fuzzy measure defined by a relation that constrains the input variable for *temperature_of_the_water* and produces a test score $m_{TEMPERATURE}(z)$. The values that *temperature_of_the_water* can take are defined by a fuzzy definition of TEMPERATURE. This definition signifies a possibility distribution of the variable *temperature_of_the_water*. How TEMPERATURE achieves this possibility distribution is programmed into the fuzzy system, either after consultation with experts in the field of application or after using some kind of learning from examples (Kosko 1992a). The values of *temperature_of_the_water* are used to define the test score $m_{TEMPERATURE}(z)$. This test score is in the interval [0,1] and its value describes the level of water temperature. It is a fuzzy interpretation of the input value for this variable. Triangular or trapezoidal membership functions map the input value into intervals in [0,1]. Trapezoidal membership functions are shown in Figure 2.3. Trapezoidal membership functions are used in exactly the same way as triangular memberships.

2.2.2 Firing a fuzzy rule.

In order to fire a fuzzy rule, the membership values for each fuzzy premise variable and linguistic label are calculated. These are the test scores for each premise in the antecedent of the rule. Next, the min or max functions are used to calculate the overall value of the antecedent. The min function is used when the fuzzy premises are

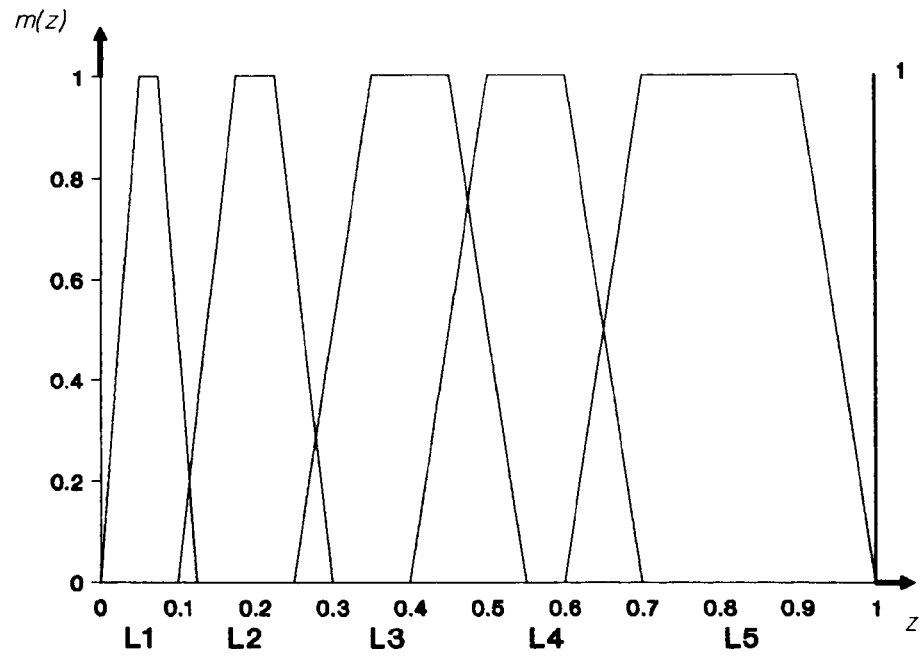

Figure 2.3 Example of trapezoidal membership functions.

linked together by the AND-connective. The max function is used for the OR-connective. In situations where estimates can vary drastically, robust fuzzy logic connectives are the least sensitive to variations in interpretation and belief. It was recently shown that min(a,b) is the most robust AND-operation and max(a,b) is the most robust OR-operation. These results were proven as a pair of theorems (Nguyen, Kreinovich and Tolbert 1993).

2.2.3 Defuzzification.

The consequent of a fuzzy rule is not evaluated until all the rules in the fuzzy ruleset are fired. Single numerical values are calculated from the membership functions as outputs of the fuzzy system. This process is called defuzzification and a number of defuzzification strategies have been developed. These include the mean of maximum-membership defuzzification method, Tsukamoto's defuzzification method,

a defuzzification method for when the output of the fuzzy rules are functions of their inputs and the centroid defuzzification method (Berenji 1992).

The mean of maximum-membership defuzzification method generates a discrete output value by averaging the support values at which the membership values reach their maximum. In the discrete case, this is calculated by:

$$Z^* = \sum_{j=1}^{k} \frac{z_j}{k}$$

where k is the number of quantized z values which reach their maximum memberships. Tsukamoto's defuzzification method calculates a discrete output value by:

$$Z^* = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

where n is the number of rules with firing strength (w_i) greater than 0 and x_i is the amount of control action recommended by rule i. This method is used for monotonic membership functions. Fuzzy rules can also be written so that the outputs are functions of their inputs. In this case, a rule i is written as:

IF $X_1 = A1_i$ AND $X_2 = A2_i$ AND ... $X_n = An_i$

THEN $Z = f_i(X_1, X_2, \ldots X_n)$

Assuming that α_i is the firing strength of i, the discrete output value is calculated by:

$$Z^* = \frac{\sum_{i=1}^{n} \alpha_i f_i(x1_i, x2_i, \ldots xn_i)}{\sum_{i=1}^{n} \alpha_i}$$

where n is the number of firing rules.

The centroid defuzzification method is the most commonly used because it gives a unique value and it uses all the information in the output distribution (Kosko 1992a, 1993b). The method works by combining all the fuzzy rules in the knowledge base that have the same fuzzy variable as consequent. Each time a fuzzy rule is fired, the support for the fuzzy premise is calculated and the area of the triangular function of the consequent up to this level of support is used to calculate the output value. Equation (2.2) is then used to derive the defuzzified value B:

$$B = \frac{\sum_{j=1}^{n} y_j \, m_B(y_j)}{\sum_{j=1}^{n} m_B(y_j)} \qquad (2.2)$$

where $m_B(y_j)$ is the level of membership at the value y_j for the fuzzy variable, and y_j partitions the range of values of the consequent variable into n equal sections (Kosko 1992a). The value of B is a discrete real value in the interval [0,1]. This value is the centre of gravity of the distribution of the output set. When more than one rule is fired, a number of areas can be included in this calculation. These areas will overlap. A simple outline of the areas results in the same output value being calculated whenever the number of overlapping areas is large. Adding the overlapping areas produces an additive output set that avoids this problem (Kosko 1993a).

As an example, take a consequent variable B and assume this variable has the following support for its term set: {L1 = 0.0, L2 = 0.0, L3 = 0.0, L4 = 0.35, L5 = 0.25}. The additive output set for this fuzzy variable is shown in Figure 2.4. The range of values for the variable B are split into a number of equal sections, say, 0.0, 0.1, 0.2, ..., 1.0. The maximum vertical value in the output set at each of these

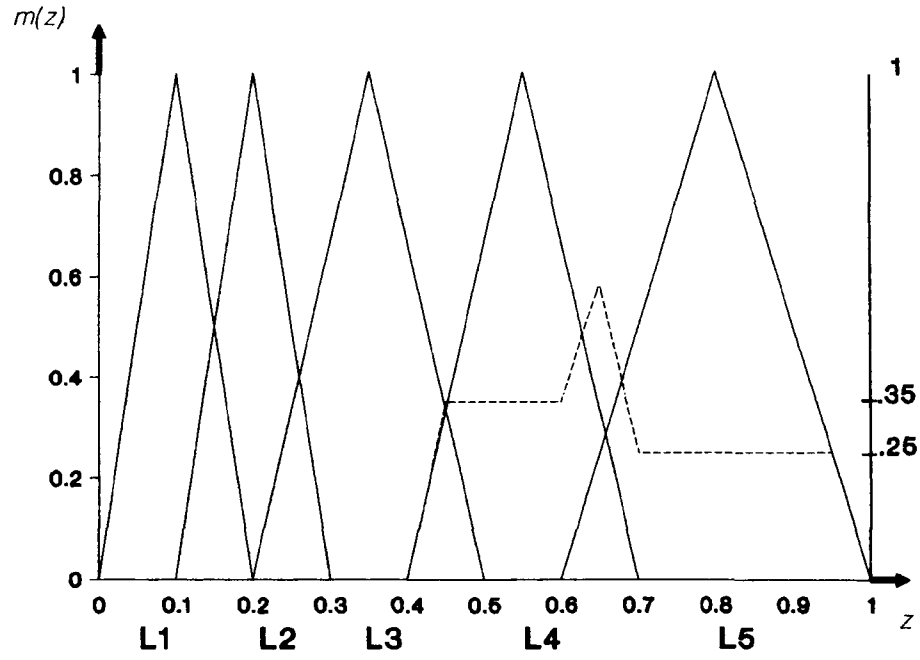

Figure 2.4. The additive output set for the fuzzy consequent variable B is the area beneath the dashed line. The area lies between the points 0.4 and 1.0 on the horizontal axis.

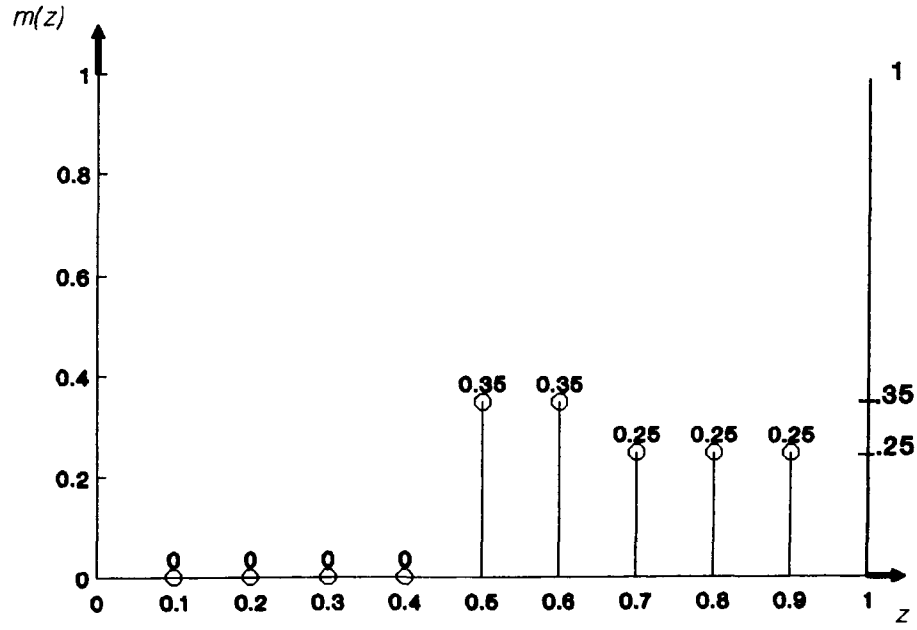

Figure 2.5. Values taken from the additive output set are used to calculate the defuzzified value of the fuzzy consequent B.

sections is taken. These values are illustrated in Figure 2.5. Equation (2.2) is then used to produce the following calculation:

$$B = \frac{0.5 \times 0.35 + 0.6 \times 0.35 + 0.7 \times 0.25 + 0.8 \times 0.25 + 0.9 \times 0.25}{0.0 + 0.0 + 0.0 + 0.0 + 0.0 + 0.35 + 0.35 + 0.25 + 0.25 + 0.25} = 0.679$$

Therefore, the result of centroid defuzzification is 0.679, a discrete value.

2.3 Fuzzy systems in changing application domains.

Kosko's Fuzzy Approximation Theorem proves that fuzzy systems adaptively estimate continuous functions from data without specifying mathematically how outputs depend on the inputs (Kosko 1992b). This theorem indicates that a set of fuzzy rules can be created so that these rules estimate the relationships between inputs and outputs to the system. There are two major constraints on these kinds of fuzzy systems. The first is that the functions estimating the relationships between inputs and outputs must be continuous. The second constraint is that the application must be non-changing or static. When the application domain is changing, the continuous function that maps inputs to outputs also changes. This means that a set of fuzzy rules that estimates one continuous function will become ineffective as new relationships between inputs and outputs occur. In fact, for the system to be effective in the context of these new inputs and outputs, the fuzzy rules that estimate the continuous function will need to change in order to adaptively estimate the new continuous function. Thus, when application domains continually change, the fuzzy rules that model these domains will also need to continually change (Partridge and Tjahjadi 1994b).

There are three ways in which a fuzzy rulebase can be changed. Firstly, the focus of attention can be shifted from one set of fuzzy rules to another. Secondly, the meaning of fuzzy rules can be fine-tuned and changed. Thirdly, new fuzzy rules can

be generated to deal with new situations and facts.

2.3.1 Shifting the focus of attention from one set of fuzzy rules to another.

By shifting the focus of attention from one set of fuzzy rules to another, attention is focused on those rules that produce the best decisions given the current inputs and outputs. The performance of the fuzzy system is optimised by choosing those rules that have been successful in the recent past.

This can be implemented by weighting each of the rules relative to each other and then altering these weights over time. The weight assigned to a particular rule depends on the past performance of that rule (Rada 1985, 1991). A training phase is used to adjust the rule weights until equilibrium is reached. In supervised training, inputs to the system are tested using perfect outputs produced after consultation with a human expert. In unsupervised training, statistical methods are used that compare the performance of each rule with the error rate of the fuzzy system. Rules which contribute to low error rates have their weights slightly amplified. Rules that contribute to high error rates have their weights reduced (Cox 1993). The main problem with these kinds of systems is that whenever the ruleset is incomplete then the determination of the correct rule weights is an NP-hard problem (Valtorta 1988). This is shown to be the case independent of the size of the rulebase.

Another method is to use connection matrices and experts' credibility weights in order to shift the focus of attention between different sets of rules (Kosko 1987). A connection matrix for an individual expert is built from the heuristics used by that expert. The different sets of fuzzy rules used by different experts need not be equivalent, but each expert is given a credibility weight. This is a measure of the experience of the expert. The individual expert's connection matrix is scalar multiplied

by his/her credibility weighting, and all the connection matrices are added together. The fuzzy system makes deductions on the basis of those fuzzy rules with the highest weightings in the combined matrix (Partridge and Tjahjadi 1991). The result of each decision made by the fuzzy system is fed back and used to adapt the credibility weights and hence shift the focus of attention between different sets of rules. A detailed presentation of this method is to be found in Chapter 4.

2.3.2 Fine-tuning and changing the meaning of fuzzy rules.

At a finer grain, the meanings of specific fuzzy rules can be fine-tuned and changed. In this way, the fuzzy rules are customised to particular types of input and output data.

One way to fine-tune fuzzy rules is to use a linguistic hedge or modifier to change the meaning of a fuzzy set. A number of modifiers have been defined, including very, more or less, plus and slightly (Zadeh 1987, Zimmermann 1991). Each of these modifiers changes the test score $m_A(z)$. For example,

$$\text{very } A = (m_A(z))^2$$

$$\text{more or less } A = (m_A(z))^{1/2}$$

These modifiers alter the shapes of membership functions and in this way they change the meanings of the linguistic labels used in fuzzy rules.

A second way is to select alternative models of defuzzification (Cox 1993, Kosko 1992a). In Section 2.2.3, four methods are described. Although the centroid method uses all the information in the output distribution and gives a unique value, it may be the case that particular input/output relationships are better modelled by an alternative method. When the relationships are modelled by monotonic membership

functions, Tsukamoto's defuzzification method will be the most appropriate. When the output values are functions of the inputs to the fuzzy system, then the defuzzification method will need to incorporate this information in its calculations. A meta-rule can be incorporated into the fuzzy system that makes decisions between different methods of defuzzification when this is appropriate.

A third way is to structurally modify the membership functions of the fuzzy premise variables. A set of meta-rules can be used to implement this. These meta-rules use performance measures defined in terms of the application domain. A scaling factor is applied to all inputs to the fuzzy system and these weights are altered on the basis of the performance measures. Thus, the scaling factor is incrementally changed until the desired output values are achieved (Daugherity et al 1992). The membership function relating to a particular linguistic label is narrowed or widened depending on whether the performance measure is above or below an expected value.

An alternative method is to use a simplified version of a fuzzy rule that uses discrete values in its premises. Each time a good decision occurs using this rule, the input value that fires the rule is recorded. These values can be used to change the premise value in order to adapt the rule more precisely to the input data (Partridge and Tjahjadi 1994a). This method is presented in detail in Chapter 5. The trouble with this method is that it fails to utilize the effectiveness of fuzzy rules that use linguistic values in their premises and consequents. These linguistic values interpret knowledge as a collection of elastic constraints on a collection of variables. This is effected by mapping these values to a set of membership functions, usually a set of triangles. Inference is then viewed as a process of propagation of elastic constraints (Zadeh 1992a). A more effective method for changing the meaning of fuzzy rules is to adapt and fine-tune the triangular membership functions attached to the term set of linguistic

values. An algorithm that alters the membership functions of fuzzy premise variables is presented in Chapter 5. This algorithm fine-tunes and changes the meanings of fuzzy rules so that the performance of the fuzzy rulebase is optimised (Partridge and Tjahjadi 1994b). The method is more fine-grained and sensitive than the method using performance measures and scaling factors. Whereas the first method simply narrows or expands the membership functions, this new method uses both the negative and positive decisions to both alter the width and move the triangles until they reach a new equilibrium. By continually applying this method, the meaning of fuzzy rules can be altered to suit current input - output data.

2.3.3 Generation of new fuzzy rules.

The third way in which a fuzzy rulebase can be changed is to create new fuzzy rules. There are three main methods used to do this: neural networks; genetic algorithms; and inductive learning. Neural networks have been used to adaptively infer new fuzzy rules from training data (Lee 1991, Kosko 1992a). An unsupervised adaptive clustering scheme will "blindly" generate and refine a bank of rules. A supervised learning technique can be used when there is additional information to accurately generate error estimates. Genetic algorithms (GAs) have also been used. This method takes an existing fuzzy ruleset and maps each rule into a set of numbers or characters, each element of which is a representation of a rule element (Johnson and Feycock 1991). Operators then act on the representation of the rule in order to incrementally create new rules. The simplest GAs use three operators: reproduction, crossover and mutation. Each new rule is checked against instances from the real world and a fitness measure is used to decide whether the rule is to be integrated into the rulebase. An important difficulty with GAs is to define an adequate fitness

function. New fuzzy rules have been created using a GA in the field of fuzzy control (Karr and Gentry 1993). The third method is inductive learning (Arunkumar and Yagneshwar 1990). Inductive learning schemes automatically generate new information by abstracting generalities from a set of instances of some phenomenon (D. Partridge 1992). In the case of a fuzzy system, the set of instances are examples of inputs and outputs to the system and the generalities abstracted are a set of new fuzzy rules. These three methods are important paradigms in the field of machine learning (Forsyth 1989, Kocabas 1991).

For completeness, mention should also be made of two other important machine learning paradigms, explanation-based learning (EBL) and case-based learning (CBL). EBL is a deductive scheme that uses a single training example in conjunction with domain knowledge to generate an operational version of an already known concept (Bergdano, Giordana and Saitta 1991). The proof is viewed as an "explanation". In the CBL paradigm, a previous case is used to solve a new problem. Cases similar to the current case are retrieved from the case memory and the most appropriate case is selected and modified to fit the current case (Nakatani, Tsukiyama and Fukuda 1991, Maher 1991). This paradigm is similar to EBL in that the strategy is stored for a particular problem. It differs from EBL in that the strategy is not simply re-used, but is actually modified. Other machine learning paradigms include conceptual clustering, learning from analogy, rote learning, advice taking, learning from examples, apprentice learning, etc., but these are usually variations of the five paradigms just described (D. Partridge 1992).

The most suitable paradigm for learning new fuzzy rules in a changing application domain is inductive learning. Neural networks are a good technique for generating and fine-tuning banks of fuzzy rules. But a changing application domain

will often need to generate just a few new rules in order to adapt to new situations and facts. By focusing in a directed manner on the most recent set of instances, and creating new rules from this small example set, inductive learning is more efficient at generating small sets of relevant rules. Genetic algorithms generate rules by mechanically changing existing rules. Then these rules are filtered through a set of fitness functions. These functions guide the search for new rules by using statistical measures. In the case of a changing application domain, it is computationally more efficient to focus on recent instances and use these to generate relevant rules rather than to syntactically alter existing rules in order to create new ones. By filtering inductively learned rules through a set of fitness functions, the statistical methods of genetic algorithms are combined with the more intentional and directed clustering techniques of inductive learning. EBL and CBL depend on stereotyped problem domains and are not really applicable in a domain that is steadily and ineluctably changed.

2.4 Inductive learning and fuzzy systems in changing application domains.

Inductive learning uses an example set of specific instances in order to infer concepts or principles. A large number of inductive learning systems have now been implemented, including INDUCE, EXTRAN7 and GLAUBER (Forsyth 1989). Most of these systems classify a training set of examples according to a predefined set of concepts or classes. For example, the ID3 algorithm of J. R. Quinlan creates a classification tree from an example set of features of domain objects. These features could be diet, size, colour and habitat and the classification could be for species of animals. The ID3 algorithm induces the optimal decision tree for classifying the example set of instances (Luger and Stubblefield 1989).

At an epistemological level, the concepts inferred by inductive learning cannot be proved correct. However, at a pragmatic level a statistical dimension cannot be ignored. According to Forsyth, the most useful hypotheses in real life are fuzzy rules of thumb already contradicted by a small number of counter examples (Forsyth 1989). These rules of thumb are not logically correct, but they have a very practical use. Such rules form what is termed commonsense knowledge. To illustrate this, take the example of the fuzzy rule for boiling an egg (see Figure 2.1). If the temperature of the water is very hot and the time of cooking is very long, then the egg will be hard. That is, except when the egg is badly cracked or when there is not enough water in the saucepan. The above rule is still very useful even though it is contradicted by the two cases cited. Most of the fuzzy rules used in fuzzy systems have this characteristic.

Another problem with inductive learning is that knowledge is not divided neatly into clearcut instances of concepts. The demarcation between concepts is often imprecise, uncertain and unreliable. To use a different term, instances can often be fuzzy. This kind of relationship between instances and concepts is very similar to the matching between inputs and premises in fuzzy rules (see Section 2.2). Therefore, it is natural to extend inductive learning to the generation of fuzzy rules. Rules can be generated from an example set of inputs and outputs to the fuzzy system. These examples will be derived from good decisions made by the fuzzy system. The new concepts that are inductively learned will be new fuzzy rules.

2.4.1 A five-step algorithm for generating fuzzy rules.

A five-step algorithm for generating fuzzy rules was developed (by Wang and Mendel 1992). This is a one-pass build-up procedure that uses an example set of

inputs and outputs to generate a set of new fuzzy rules. Fuzzy rules are generated as follows:

1. Divide the input and output spaces into fuzzy regions and assign each region a fuzzy membership function;
2. Generate fuzzy rules from given data pairs by assigning given inputs or outputs to the region with the maximum degree of membership. Obtain one rule from one pair of desired input-output data;
3. Assign a degree to each rule and accept only the rule from a conflicting group of rules that has the highest degree;
4. Create a combined fuzzy rulebase that covers all the possibilities in the control space, given linguistic values for each variable;
5. Determine a mapping based on the combined fuzzy rulebase by using the centroid defuzzification method.

The algorithm is applied to control systems, but is typically used in non-changing application domains. To this extent, the algorithm learns a fuzzy ruleset and then the control system uses this ruleset to calculate output values from input values in the control environment. In the case of a changing application domain, new fuzzy rules will need to be created throughout the life of the fuzzy system.

2.4.2 Generating new fuzzy rules in a changing application domain.

An inductive learning algorithm that generates new fuzzy rules in a changing application domain must focus in a directed manner on recent instances and use these to generate the new rules. This algorithm is not a one-pass build-up procedure. A new inductive learning algorithm that gathers examples from recent instances of inputs and

outputs to the fuzzy system and uses these to generate new rules is presented in Chapter 6.

2.4.3 Fitness functions for evaluating new rules.

New fuzzy rules must be tested before they are incorporated into the fuzzy system. These tests must be both rigorous and based on the correct evaluation criteria. Fitness functions are mathematical formalisations of evaluation criteria for intelligent systems. A number of fitness functions have already been implemented. The J measure (by Smyth and Goodman 1992) is a fitness function that uses a preference measure to rank new rules and choose the *K* best ones from the set. This preference measure is based on conditional probability and uses the simplicity of the hypothesis and the goodness-of-fit between the hypothesis and the data as its main criteria. The probability of the rule premise occurring is used as the measure of simplicity. Then, given this, a calculation is made of the probability of the implication. Once the new rules are induced, the preference measure is calculated for each rule and the *K* most informative rules are accepted into the rulebase. In another system (by Gaines and Shaw 1986), subjective repertory grid numbers are input by a human expert and these are transformed into fuzzy logic parameters which, in turn, are used to obtain a measure of the information content of the associated rules. A fitness measure is used which is based on "surprise minimisation," defined as minus the log of the probability ascribed to that event before it occurred.

But these systems have concentrated on just one or two criteria. Six fitness functions based on six well-established evaluation criteria for knowledge-based systems are presented in Chapter 6.

CHAPTER 3. Review of X-ray rocking curve analysis.

3.1 Introduction.

X-ray rocking curve analysis is an example of a changing application domain. An X-ray rocking curve is the angular reflectivity plot obtained when a crystal is rotated in an X-ray beam through an angle at which it diffracts strongly (Halliwell, Lyons and Hill 1984). This curve is highly sensitive to the details of the structure of the outer layers of the crystal. X-ray rocking curve analysis is widely used in industry and research to investigate the quality of natural and synthetic crystals (Bowen, Hill and Tanner 1987).

Analysing a rocking curve means inferring a structural description of the crystal from particular features of the curve. But it is difficult to perform such an analysis. The intensity of the X-ray beam is measurable, but the phase is not. The latter is needed for a direct reconstruction of the structure from the curve. Therefore, the usual method of analysis is to infer a trial structural description and use this to calculate a theoretical curve. The theoretical curve is compared with the experimental one and the structural description is fine-tuned. New theoretical curves are calculated and this process continues until the theoretical curve closely matches the experimental one. At this point, an adequate structural description of the crystal has been formulated.

Eight basic types of structures are analysed: a substrate only; a single layer; a single graded layer; multiple layers; multiple graded layers; Multiple Quantum Wells (MQWs); superlattices; and superlattices with additional layers top and bottom. Each of these structures has a characteristic rocking curve. The shape of a particular curve can deviate from the characteristic curve because of specific structural factors.

For example, the shape of a layer peak can be altered because of grading or a peak can be broadened because of curvature of the specimen.

A human expert usually begins analysis by describing the major features of the curve. Following this, the expert deduces relationships between these descriptive parameters and the structural parameters. This is the difficult part of X-ray rocking curve analysis. Sometimes these inferences are straightforward, but often the inferences involve making vague deductions about imprecise and uncertain concepts. But more importantly, deductions made about one type of structure may be incorrect and useless for another type. The number of possible structures and crystals in X-ray rocking curve analysis is infinite and it is impossible to prescribe for all these possibilities in a fixed set of rules.

There are two reasons for formalising the analysis of X-ray rocking curves. Firstly, there are few experts who can perform this type of analysis and it would be an advantage to develop an intelligent system for this task. Secondly, the novice takes up to 50 iterations (as against 5 - 10 for the expert) to produce good theoretical curves. An intelligent system that makes good initial deductions about structural parameters will greatly increase the speed of analysis by the novice user. For these two reasons, an intelligent system has been developed for this application (Partridge and Tjahjadi 1994a 1994b).

When the heuristics of experienced experts are translated into rules, it is found that these rules often relate imprecise and uncertain concepts. Therefore, fuzzy rules are used to model the relationships between descriptive parameters and structural parameters in X-ray rocking curve analysis. But the relationships between these parameters can change over time. Self-adaptation and rule generation are used to

model these changes.

3.2 X-ray rocking curves.

To generate an X-ray rocking curve, a fixed crystal called the reference crystal is set up to reflect monochromatic (Kα) radiation in the direction of the crystal to be examined. The latter crystal is then rotated through the Bragg angle θ, while the beam reflected by it is measured in a fixed counter with a wide slit. The curve of intensity of the beam versus angle θ is called a rocking curve, and the instrument itself is called a double-crystal diffractometer (Cullity 1978, Hart 1980). This experimental arrangement is shown in Figure 3.1.

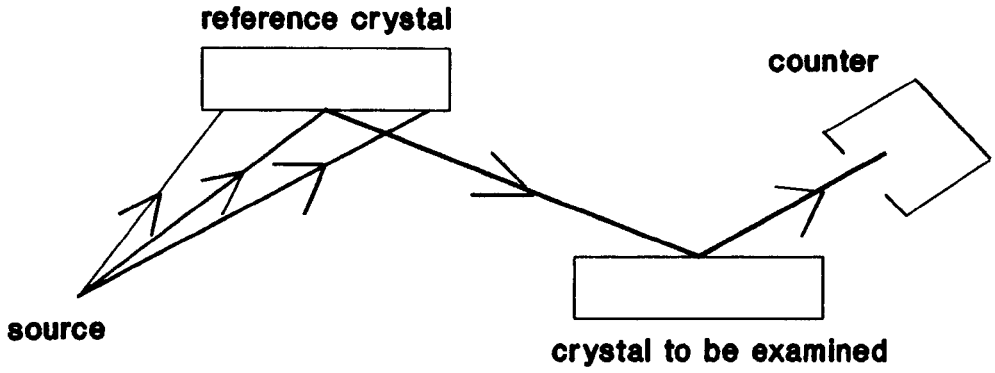

Figure 3.1 A double-crystal diffractometer for generating X-ray rocking curves.

As the crystal is rotated, the diffracted intensity changes. The relative strength of the direct and diffracted beams emerging from the crystal will depend on how much the angle between the incident beam and the diffracting planes deviates away from the Bragg angle θ. The widths of rocking curves are a function of the material, the reflection and the wavelength band selected by the reference crystal. They are typically 5 - 10 arc seconds wide for strong reflections. The important point about

Bragg reflection is that a condition must be satisfied for maxima of diffracted intensity to occur. This condition is called Bragg's Law and is written:

$$n\lambda = 2d \sin \theta$$

where n is a constant, λ is the wavelength, d the spacing of the diffracting planes and θ the angle of incidence of the X-ray beam on the diffracting planes (Cullity 1978, Bede 1992). Examples of rocking curves are shown in Figure 3.2 and Figures 3.4 to 3.10.

3.3 Deducing structural parameters from an X-ray rocking curve.

When semiconductor crystals are grown, they are built up of a substrate of one material and layers of other materials. It is important for the crystal grower to be able to identify the composition of these layers. But it is also important to identify a number of other parameters: the thickness of the layers; the lattice mismatch; grading; tilt; roughness between layers or between layers and substrate; relaxation; curvature; period of the superlattice; period dispersion etc. These parameters are called structural parameters and they provide a mechanism for describing the structure of the crystal, which affects the electronic properties (Bowen, Hill and Tanner 1987, Fewster and Curling 1987)).

It is not easy to derive the structural parameters from the rocking curve. This is because, in producing a rocking curve for a grown crystal, the intensity of the X-ray beam is measurable but the phase is not. The phase is needed for a direct reconstruction of the structure from the rocking curve. However, if the structure of a crystal is known then the rocking curve for that crystal can be simulated.

3.4 Simulating an X-ray rocking curve.

Simulating an X-ray rocking curve involves solving the Takagi-Taupin equations, a coupled pair of linear first-order differential equations, while making allowances for the structural parameters. These equations give the rate of change of the ratio of diffracted to incident beam amplitudes as a function of depth below the surface (Halliwell, Lyons and Hill 1984). The basic Takagi-Taupin equations are:

$$\frac{i\lambda\gamma_H}{\pi}\frac{dD_H}{dZ} = \psi_0 D_H + c\,\psi_H D_0 - \alpha_H(\omega) D_H$$

$$\frac{i\lambda\gamma_0}{\pi}\frac{dD_0}{dZ} = \psi_0 D_0 + c\,\psi_H D_H \qquad (3.1)$$

where

D_0 is the incident beam amplitude;

D_H is the diffracted beam amplitude;

Z is the depth into the crystal;

$\gamma_H = n.k_H$ where n is the surface normal; and

k is the diffracted beam vector;

λ is the wavelength;

c is the polarisation factor (= 1 or $|\cos 2\theta|$);

ψ_H is related to the structure factor F_H by

$$\psi_H = (\lambda^2 r_e / \pi v) F_H$$

where v is the unit cell volume; and

r_e is the electron radius (Fewster and Curling 1987).

Simulation programs like RADS (RADS 1992) use equations (3.1) to simulate theoretical rocking curves.

3.5 X-ray rocking curve analysis.

The method of X-ray rocking curve analysis is to describe the structure of the crystal in reasonably precise terms, simulate a rocking curve for that crystal, and compare it with the experimental rocking curve. The first description of the structure will be redefined on the basis of this comparison. This will be used to simulate another curve. This process is repeated until the simulated rocking curves gradually approach the shape of the experimental curve. When the matches are close, then it can be said that an adequate structural description of the crystal has been formulated.

In general, the curves will have a peak for the substrate, one or more peaks for the layers, and a number of satellite peaks. Although a peak can be formed by an individual layer and in principle a layer will give rise to a peak, peaks can also be due to interference between layers. These interferences can be either positive or negative. Positive interference will result in additional peaks, while negative interference can cancel out a peak. For this reason, formalising the analysis of X-ray rocking curves is more difficult than might at first be expected.

X-ray rocking curve analysis is also a changing application domain. The types of structures analysed change over time and the relationships between rocking curves and structural parameters will be different for these different structures. The number of possible structures and crystals in X-ray rocking curve analysis is computationally infinite and it is impossible to prescribe for all these possibilities in a fixed set of rules. However, there are eight types of structures that have been identified. These are structures that produce characteristic rocking curves that have some generic features.

3.6 Structures analysed in X-ray rocking curve analysis.

The structures analysed in X-ray rocking curve analysis can be split into eight basic types: a substrate only; a single layer; a graded single layer; multiple layers; graded multiple layers; Multiple Quantum Wells; superlattices; and superlattices with a number of additional layers top and bottom. From the interviews conducted with the domain experts Prof. B. K. Tanner (Henson 1993), Prof. D. K. Bowen, Dr. N. Loxley (Henson 1993) and Dr. C. R. Thomas, the characteristic rocking curves produced by these structures were noted.

3.6.1 A substrate only structure.

A substrate only structure is characterised by one peak only. An intrinsically narrow curve is expected when the crystal is effectively perfect. The full width at half maximum (FWHM) depends on the material. The height of the peak depends on the X-ray source. The longer the wavelength of the X-ray source then the wider the peak. E.g. the FWHM for Si is 3.5 arc seconds for the 004 symmetric reflection with CuKα radiation (1.541 Å) (RADS 1992). This curve is shown in Figure 3.2.

In a substrate only structure, defects will lead to a broadening of the curve. Therefore, the width of the curve is used as a measure of lattice perfection. In most cases, the total integrated intensity under the curve remains the same.

In the case where the experiment is set up with a reference crystal that is a different type to the substrate, then there can be two peaks. These two peaks have the characteristic feature that they are usually in the ratio of 1 : 2 and the separation of the two peaks can be calculated using Bragg's Law.

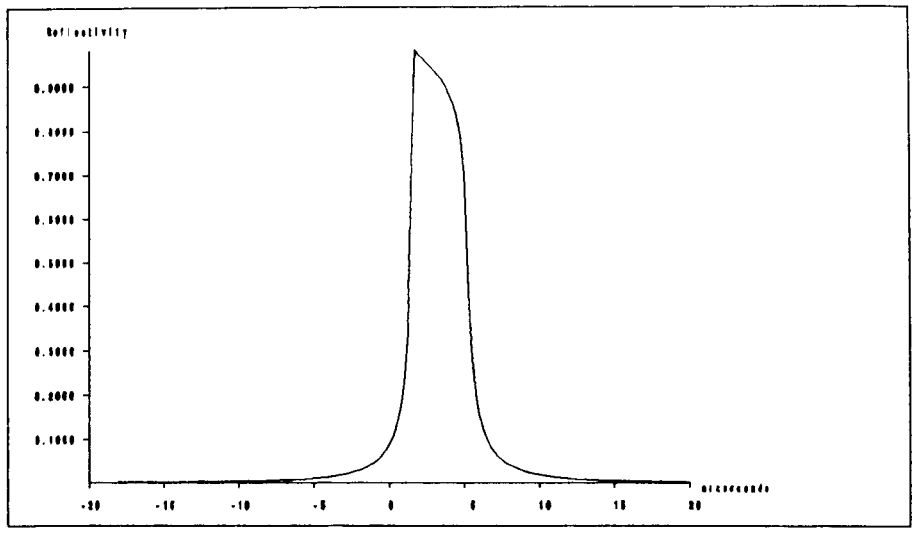

Figure 3.2 Example of a rocking curve for a Silicon substrate only structure. The peak is lower on the high angle side because peak reflectivity is not 100% due to absorption (RADS 1992).

3.6.2 A single layer structure.

A single layer structure is characterised by two peaks if the thickness of the layer is between 0.5 μm and 5 μm and the layer has a different lattice parameter to the substrate. The lower limit for layer thickness is given by interference effects and the upper limit by extinction distance, but the exact limits depend on the particular structure being measured. If there is no layer peak, this indicates that the layer is either very thin or it is mismatched by a larger amount than expected, and the peak is therefore outside the scan range.

Generally, the substrate and layer peaks can be labelled and their separation determined (Fewster and Curling 1987). The distance between these peaks depends on what is grown. The angular separation of the peaks $\Delta\omega$ can be related to the lattice

mismatch through the differential form of Bragg's Law:

$$\left(\frac{\Delta a}{a}\right)_{\perp} = - \delta\theta \text{ Cot } \theta \qquad (3.2)$$

where $\delta\theta$ is the peak splitting and θ is the Bragg angle. The experimental mismatch $(\Delta a / a)_{\perp}$ is related to the relaxed mismatch by:

$$\left(\frac{\Delta a}{a}\right)_{r} = \left(\frac{\Delta a}{a}\right)_{\perp} \left(\frac{1 - \nu}{1 + \nu}\right) \qquad (3.3)$$

where ν is the Poisson ratio of the material. For a thin layer (less than 1 - 2 μm thick) on a closely matched substrate, the angular separation may not give the true mismatch. This is due to the influence of the beam entering the substrate. Only through simulation can the true value be derived (Fewster and Curling 1987).

If there are two peaks, but the intensity is very much lower than expected then this indicates that the rocking curve of the layer is broadened and the epitaxy is not good, i.e. there are lots of defects on the interface and possibly defects in the layer. If a mismatched layer is grown on a substrate, the substrate assumes an overall curvature which increases as layer thickness and layer mismatch increase. This also results in broadening of a peak (Halliwell, Lyons and Hill 1984). This broadening is normally symmetrical, unlike the asymmetrical broadening associated with grading (see Section 3.6.3). In a symmetric geometry, the substrate peak can be expected to be symmetric and asymmetry in this peak can indicate that there is a layer present in the peak.

Interference fringes can appear on the sides of the layer and substrate peaks. If there is no grading the separation of these fringes may be used to determine the thickness of the layer. The layer thickness is calculated using the equation:

$$\text{thickness of layer} = \frac{\lambda \gamma_i}{\delta\theta_p \text{ Sin } 2\theta} \qquad (3.4)$$

where $\delta\theta_p$ is the spacing of the interference fringes, θ is the Bragg angle and γ_h = Cos (θ+90°). Fringe spacing is inversely proportional to layer thickness and as the layer becomes thinner, the thickness fringe oscillation period increases, the height of the peak is reduced and the FWHM of the peak increases in width (RADS 1992). The peak corresponding to the layer moves closer to the substrate peak as the layer gets thinner. The physical reason for this is not fully understood. Thus, the separation between the layer and substrate peaks is a function of the thickness of the layer. For very thin layers, peak separation cannot be used to determine mismatch and simulation must be used instead. If the layer and substrate peaks do not overlap, values can be derived for the FWHM and the ratio of layer to substrate peak areas (the integrated intensity ratio) as a function of layer thickness. The integrated intensity ratio can also be used to measure layer thicknesses for good quality layers. The method is to graph the integrated intensity ratio against the layer thickness for a particular substrate and layer. The graph is linear up to approximately a micron thickness. For larger thicknesses a piece-wise linear model is used. This method is especially useful if there is poor visibility of interference fringes. An example graph is shown in Figure 3.3.

If the layer is tilted relative to the substrate, this produces a shift in the position of the layer peak relative to the substrate peak. This shift is unconnected with composition and in order to distinguish the tilt from the true mismatch, the specimen should be rotated about its normal and at least three further measurements taken.

The layer peak is generally wider than the substrate peak, though this is not always the case. If there is large mismatch, the tilts and defects will give rise to a broadening of the layer peak. There will be large separation between the layer peak and the substrate peak as well. Broadening of the substrate peak will indicate bending

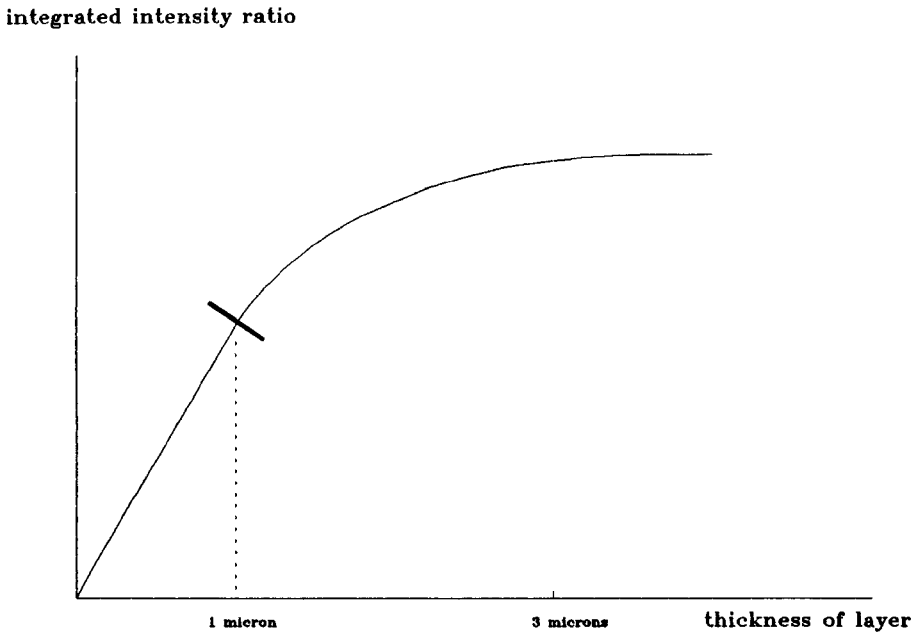

Figure 3.3 Graph of integrated intensity ratio against layer thickness. The values must be measured for particular substrates and layers.

of the substrate. Relaxation can also be expected in a layer with large mismatch.

Relaxation is the extent to which the interface is less than perfectly coherent with the substrate. The equation (3.3) for relaxed mismatch assumes that the interface is fully coherent. If there are interface distortions it is important to measure the misfit parallel as well as perpendicular to the interface. To derive this measurement, an asymmetric reflection which is at as high an angle to the surface as possible is needed, e.g. 224 and 511 (where these values are the Miller indices of the diffracting planes multiplied by the order of diffraction) (Bede 1992).

An example of a rocking curve for a single layer structure is shown in Figure 3.4.

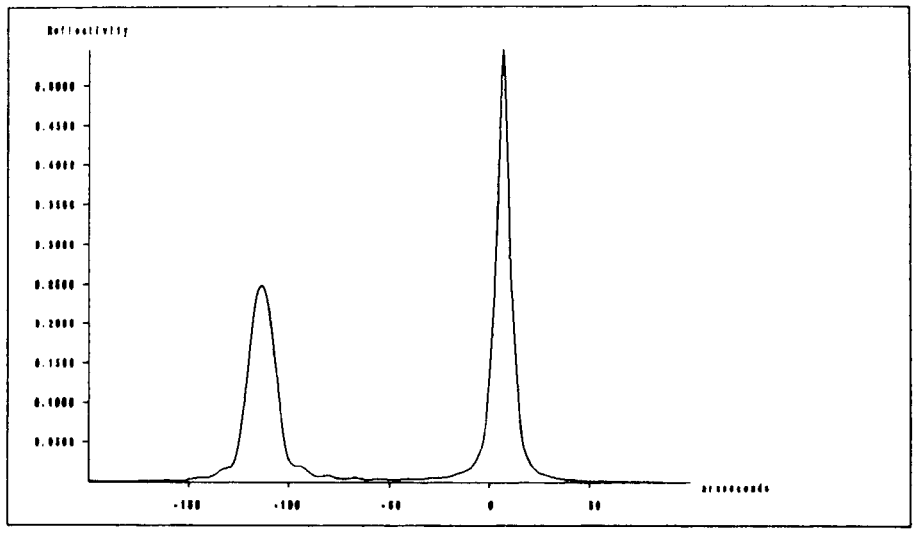

Figure 3.4 Example of a rocking curve for a single layer structure. The curve shows a 1.45 μm thick layer of $Al_{0.301}Ga_{0.699}As$ grown on a GaAs substrate (RADS 1992).

3.6.3 A single graded layer structure.

Grading means micro-changes in the lengths and angles of the unit cell of the lattice. If the lattice parameter is varying with depth, the layer peak is asymmetric in shape. It is also wedge shaped. Grading occurs only on nearly matched layers, i.e. the lattice parameters are close. Near matching is indicated by the closeness of the peaks.

Small changes in the unit cell of the lattice produces a layer peak that is asymmetrical. As the mismatch gradient increases, the layer peak will develop shoulders and two distinct maxima will appear. With further increase in the mismatch gradient, a distinctive wedge shape appears that has several maxima (Halliwell, Lyons and Hill 1984). The direction of the wedging depends on the composition, but there are exceptions to this. In some cases, the mismatch where the grading starts and ends

can be calculated from the end points of the wedge structure. But this is not always the case. Visibility of interference fringes is reduced when there is grading. This affects the calculation of the layer thickness. An example of a rocking curve for a graded single layer is shown in Figure 3.5.

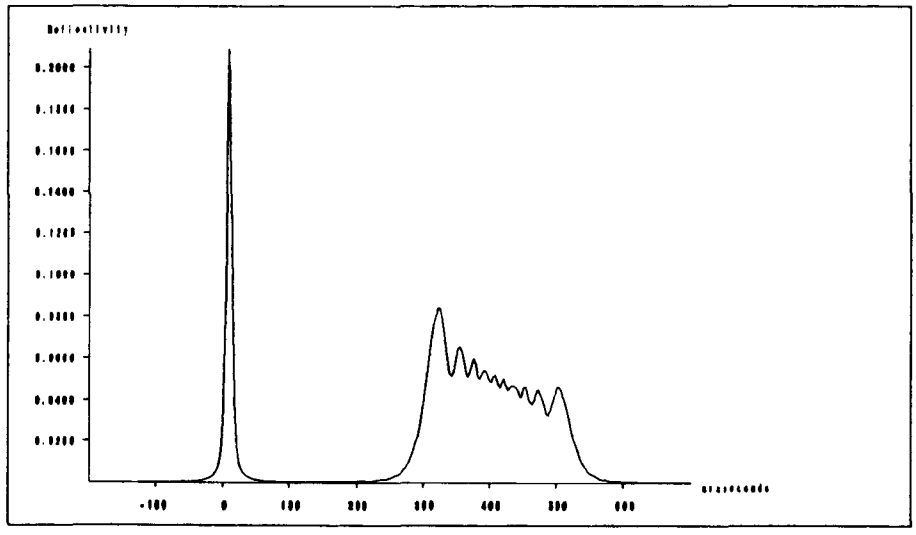

Figure 3.5 Example of a rocking curve for a graded single layer structure. The curve shows a 3 μm thick layer of InGaAs that varies from $In_{0.502}Ga_{0.498}As$ at the InP substrate interface to $In_{0.516}Ga_{0.484}As$ at the surface (RADS 1992).

3.6.4 Multiple layers structure.

For a multiple layers structure, sometimes one peak can be identified for each layer. This occurs if the layers are thick and are of different composition. For three or more layers the number of peaks in the rocking curve may or may not correspond to the number of layers, but most usually it does not.

A pair of mismatched layers will produce a rocking curve with one peak for the substrate and one for each mismatched layer (Halliwell, Lyons and Hill 1984).

With each additional layer, first the number of peaks increases and then it decreases. If there is a large number of layers, a single peak is observed for the layers. The position of this peak corresponds to the average mismatch of the stack of layers.

There are a number of interference fringe systems in a multiple layers structure. These can be difficult to separate from each other. For thin layers, the interference fringes start interfering with themselves and these interferences can get quite complicated. If the layers are very thin, then the interference effects result in a shifting of the peaks away from the expected positions. The interferences can also result in splitting of the main peaks, and it can be difficult to determine the exact number of layers and the mismatch. A lot of simulation may be necessary before these parameters can be determined.

Fourier transform can be used to derive the layer thickness from the periodicity of fringes. These fringe periods are independent of the mismatch of the layers. Fringe periods can therefore be used to measure layer thickness even when the layer and substrate peaks cannot be separated in the curve.

If there are two layers of equal composition sandwiching a fairly thin layer, this structure is called an interferometer (Tanner 1993). The interference effects in an interferometer can result in a single peak split in two. A number of strong and characteristic interference systems appear. Each layer in the structure produces a specific periodicity within the interference system.

There is no abrupt transition between a multiple layers structure and a multilayer structure or superlattice. An example of a rocking curve for a multiple layers structure is shown in Figure 3.6. On a linear scale this example appears to contain only the substrate peak. However, when the curve is plotted on a logarithmic

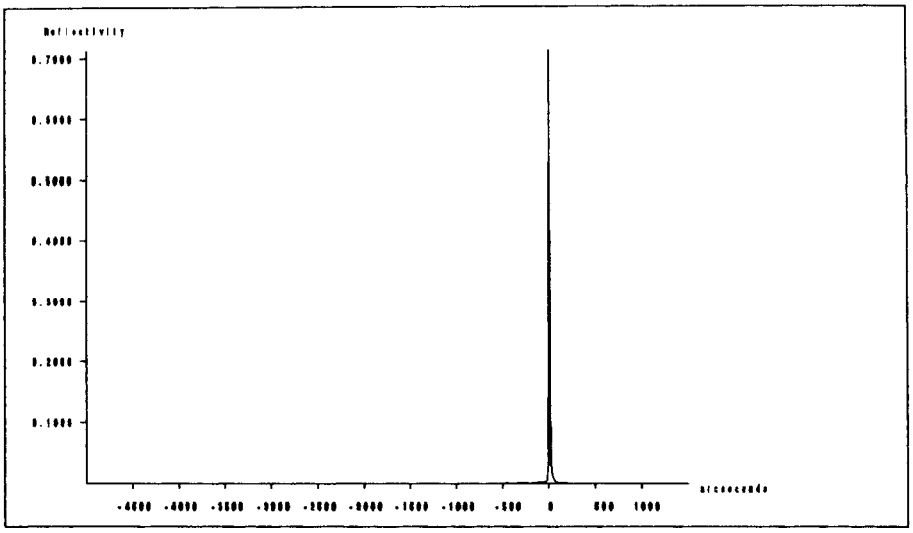

Figure 3.6 Example of a rocking curve for a multiple layers structure. The curve shows a very thin layer of 17 nm $In_{0.18}Ga_{0.82}As$ on GaAs capped with a 0.1 μm layer of GaAs (RADS 1992).

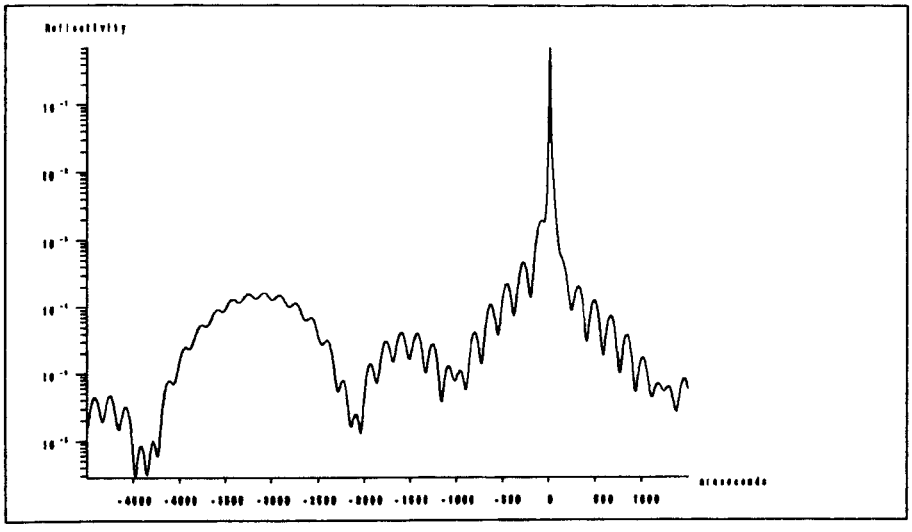

Figure 3.7 Logarithmic plot of the example rocking curve for a multiple layers structure (RADS 1992).

scale then the low intensity peak for the InGaAs layer is seen clearly. This logarithmic plot is shown in Figure 3.7.

3.6.5 Graded multiple layers structure.

Graded multiple layers show characteristic wedge-shaped rocking curves. In some cases it is possible to recognise the peak due to a uniform layer within the graded system. However, there are lots of examples where there appear to be extra layers, but the confusing oscillations are really a result of grading. For example, if part of the graded layer has the same mismatch as another layer or the substrate, then the interference effects can be very pronounced. That part of the graded layer can then behave like another very thin layer. Considerable skill is needed in proposing the first trial structure for simulation.

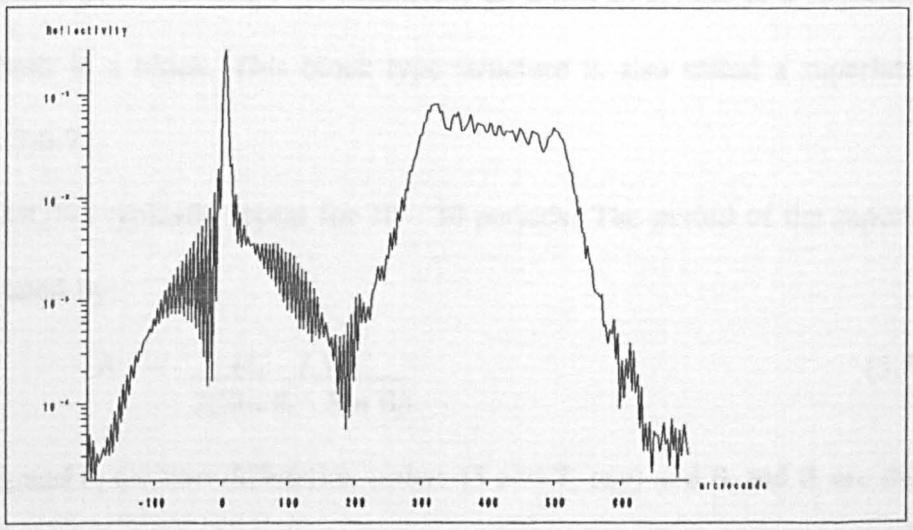

Figure 3.8 Example of a rocking curve for a graded multiple layer structure. This is a logarithmic plot of a 3 μm graded layer of InGaAs on a substrate of InP. There is also a 0.1 μm cap of InP.

The position of interference fringes depends on the mismatch and graded layers result in poor visibility of interference patterns. Thus, it is difficult to calculate the layer thicknesses for graded multiple layers. An example of a rocking curve for graded multiple layers is shown in Figure 3.8.

3.6.6 A Multiple Quantum Well (MQW) structure.

A Multiple Quantum Well (MQW) structure consists of alternating layers of different compositions. These are called the well and the barrier. The structure is composed of a substrate material A and a MQW with a stack of AB layers, where B is an alloy. This results in a rocking curve that is characterised by little satellite peaks which are equally spaced about a zero-order peak. The zero-order peak corresponds to the average composition of the AB sequence. This peak is usually displaced about the substrate peak. An MQW is effectively an artificial crystal of a sequence of AB layers built in a block. This block type structure is also called a superlattice (see Section 3.6.7).

MQWs typically repeat for 10 - 30 periods. The period of the superlattice Λ is calculated by:

$$\Lambda = \frac{(L_i - L_j)\,\lambda}{2(\sin\theta_i - \sin\theta_j)} \qquad (3.5)$$

where L_i and L_j are two diffraction orders (5 and 7, say) and θ_i and θ_j are the angles at which these orders diffract. These values are measured from any two satellite peak positions. The average of several satellites can give a more accurate result.

The spatial period of the structure changes the separation of the satellite peaks. The relative spacing of the two layers changes the relative intensity of the successive satellites. The lattice parameters affect the intensity of satellites. If there is significant

broadening of the higher order satellites, then there can be dispersion in the layer thickness or grading. This is called the period dispersion and is calculated by:

$$\text{period dispersion} = \frac{(L_i - L_j)\ \lambda\ \delta\Delta\theta}{\text{Cos}\ \theta\ \Delta\theta^2} \qquad (3.6)$$

where $\delta\Delta\theta$ is the satellite FWHM and $\Delta\theta = |\theta_i - \theta_j|$.

Very poor satellite visibility indicates that there is grading through the layers. Asymmetry in the intensity of plus and minus satellites also indicates layer grading. But it is impossible to tell from the rocking curve the difference between roughness and grading.

The same method is used to calculate mismatch as for the single layer structure (see Section 3.6.2). In this case, however, the zero-order, or average mismatch, peak is used instead of the layer peak. Equations (3.2) and (3.3) are valid only if the total thickness of the alloy (B) is greater than 0.5 μm thick (approximately). If the layers are thinner, the average layer peak is moved towards the substrate peak due to interference effects. This results in incorrect calculations from these equations. Repeated simulations must be used to derive the correct results.

An example of a rocking curve for a MQW structure is shown in Figure 3.9. This curve shows a zero-order peak associated with the layer. This peak corresponds to the average composition of the layers. When plotted on a logarithmic scale the weak satellite peaks appear, as shown in Figure 3.10.

3.6.7 A superlattice.

The MQW structure is a superlattice, but there can be superlattices with more than two layers in the sequence, i.e. ABC... These structures are also characterised by two peaks and the satellites are evenly spaced about the zero-order peak. The main

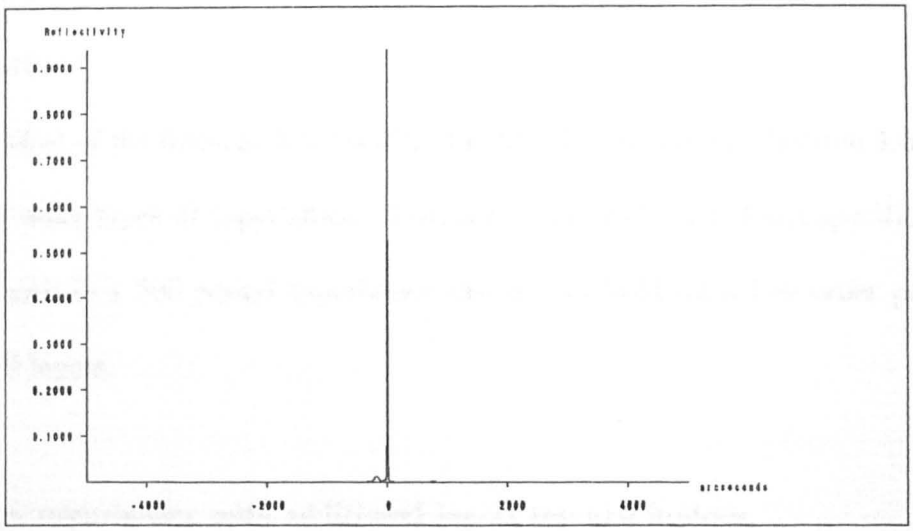

Figure 3.9 An Example of a rocking curve for an MQW structure of a 10 repeat of 10 nm AlAs and 10 nm GaAs on a GaAs substrate (RADS 1992).

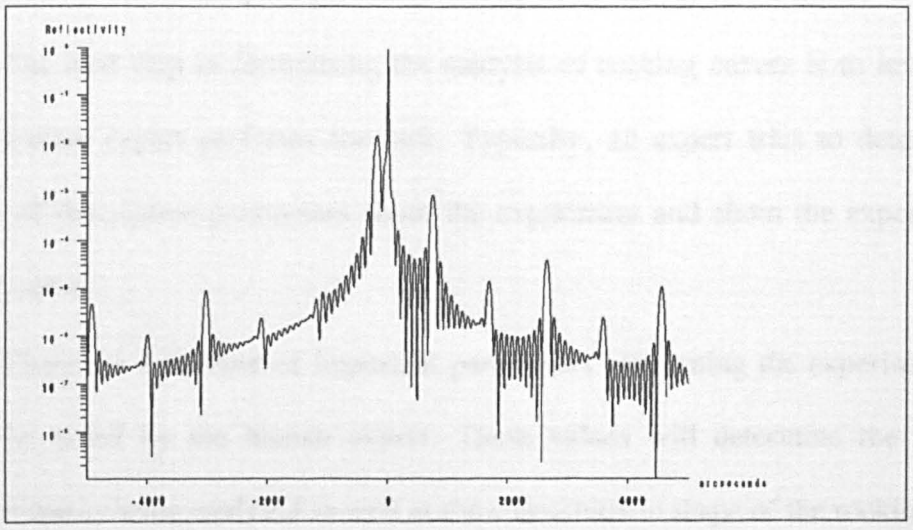

Figure 3.10 Logarithmic plot of the example of a rocking curve for an MQW structure (RADS 1992).

characteristic of a superlattice structure is a stack of layers of alternating compositions.

Most of the features described for the MQW structure (see Section 3.6.6) will hold for other types of superlattice. However, it should be noted that specific effects can be seen in a 500 period superlattice that do not hold for a low order period of about 20 layers.

3.6.8 A superlattice with additional layers top and bottom.

A superlattice with additional layers top and bottom of the block can be characterised by additional peaks associated with each of the layers. If the layers are very thin, there can be interference fringes associated with them.

3.7 How a human expert performs X-ray rocking curve analysis.

The next step in formalising the analysis of rocking curves is to investigate how a human expert performs the task. Typically, an expert tries to determine a number of descriptive parameters about the experiment and about the experimental rocking curve.

There are a number of important parameters concerning the experiment that should be noted by the human expert. These values will determine the type of structure that is being analysed as well as the characteristic shape of the rocking curve (From interview with Prof. B. K. Tanner (BKT)). These parameters include: the wavelength of the X-ray beam, which is used in a number of calculations (see Equations (3.4), (3.5) and (3.6)); the reflection indices; the substrate material; the reflection orientation (symmetric or asymmetric); the surface normal indices; the scan

range; the layer material(s); the estimated thickness of the layer(s); the lattice parameters; and the Poisson ratio, which is used in the calculation of relaxed mismatch (see Equation (3.3)).

Analysing a rocking curve is a difficult task that relies more on skill and experience than principles (From interview with Prof. D. K. Bowen (DKB)). It is important to deduce all the important structural parameters because structural features well below 10 nm in size can have a significant influence on the rocking curve. There is a large number of variables that can contribute to the definition of a curve. The first descriptive parameter a human expert will seek to identify is the number of peaks (From interviews with BKT and DKB). It is important to interpret which peaks are significant and identify the positions of these peaks. Significant peaks are those associated with the substrate, with a layer or with a stack of layers. The positions of the peaks will be used to determine the separation between peaks and hence to calculate the lattice mismatch (see Equations (3.2) and (3.3)). In fact, for a 004 reflection, 1 arc second will roughly correspond to 4 ppm and the mismatch can be estimated from this (From interview with BKT).

The heights of the peaks in relation to one another is the next important parameter. However, the heights of peaks depends on a number of factors including the intensity of the X-ray beam. Therefore, a more useful parameter is the integrated intensity beneath the peaks (From interview with DKB). This value remains constant even as the peak is broadened due to curvature or grading (see Sections 3.6.2 and 3.6.3). The integrated intensity ratio is useful in estimating the thickness of a layer.

It is important to determine the shape of the peaks in order to check for asymmetry and wedging. Asymmetry can give an indication of grading or that there

is a layer present in the substrate peak. Wedging will also indicate grading. A further parameter to identify is the existence of interference fringes. Interference effects can be related to layer thickness and low visibility of fringes may indicate that there is grading of a layer. Layer thickness can be calculated from the spacing between fringes (see Equation (3.4)). For a superlattice structure, it is important to identify the satellite peaks. These peaks should be evenly spaced about the zero-order peak (From interview with BKT).The relative spacing between satellites is used to calculate the period of the superlattice (see Equation (3.5)) and satellite FWHMs are also used to calculate the period dispersion (see Equation 3.6)).

It is also important to determine the background levels between peaks. One method is to average the intensities of the first or last few points on the rocking curve (From interview with DKB). A high background between peaks is usually a good indication that the structure is not simple.

From these descriptive parameters evidence is gathered for the structural parameters. Some structural parameters can be deduced directly from the rocking curve; some are calculated using equations, e.g. see Equations (3.3) and (3.4) for mismatch; some are deduced using vague and imprecise interpretations of the descriptive parameters; and some can only be deduced after further simulations or experiments. The structural parameters are used to simulate a theoretical rocking curve that is then compared with the experimental curve.

Some typical structural parameters for the single layer structure are as follows: mismatch, this can be deduced using the peak separation and Equations (3.3) and (3.4); crystal quality, this can be deduced from a number of factors including the number of peaks and the broadness and asymmetry in the peaks; bending of the

substrate, this can be deduced from broadening of the substrate peak; the layer thickness, this can be deduced from the spacing of the interference fringes and Equation (3.4) or using the integrated intensity ratio; grading of a layer, this can be deduced from the amount or asymmetry or wedging in the layer peak; and a layer is present in the substrate peak, this can be deduced from asymmetry in the substrate peak. These relationships, between structural parameters and descriptive parameters derived from the experimental curve, can be formulated as a set of rules. In this way, the reasoning processes of the human experts can be translated into a set of deductions in a machine-usable form.

3.8 A fuzzy system for X-ray rocking curve analysis.

There are few experienced experts in the interpretation of X-ray rocking curves and so it would be an advantage to develop an intelligent system for this application. Such an intelligent system would describe the experimental curve in reasonably precise terms, deduce the structural parameters from this curve and then propose a structure for simulation. After simulation, the theoretical curve will also be described by the system and the two curves will be compared. By doing this, an objective measure of the success rate of the intelligent system can be calculated for each decision.

Many of the descriptive parameters used to describe experimental curves are imprecise and uncertain. These are parameters like broadening of the peak, asymmetry in the peak and visibility of interference fringes. Vague concepts like these are typically modelled by fuzzy variables. However, there are descriptive parameters that are precise. These include the number of peaks and splitting of the peak. These concepts are best modelled by Boolean variables. The structural parameters have a

degree of uncertainty attached to them. They are vague to the extent that they are inferred from the rocking curve and these inferences are not always exact. Therefore, fuzzy variables can also be used to model these parameters.

The relationships between descriptive and structural parameters can be modelled as a collection of fuzzy rules. The descriptive parameters will be the inputs to the fuzzy system, the structural parameters will be the outputs. A fuzzy system for this application has been implemented and this system is presented in detail in Chapter 7. X-ray rocking curve analysis is a changing application domain, and therefore the system also includes self-adaptation and rule generation. Two methods of self-adaptation are presented in Chapters 4 and 5 and an inductive learning algorithm for generating new fuzzy rules is presented in Chapter 6. The fuzzy system for X-ray rocking curve analysis is used to test each of these techniques.

CHAPTER 4. Changing the focus of attention in a fuzzy system using connection matrices, credibility weights and Unassert/Assert functions.

4.1 Introduction.

In this chapter, a technique for changing the focus of attention between different sets of fuzzy rules is presented. The technique evaluates a set of experts' credibility weights and incrementally alters these weights to suit the most recent decisions made by the fuzzy system (Partridge and Tjahjadi 1994a). The purpose of this technique is to choose those rules that produce the best decisions given the most recent inputs and outputs.

Connection matrices and credibility weights are used to combine the heuristics of several experts in a fuzzy system (Kosko 1987). The knowledge from different experts need not be equivalent and rules from different experts can contradict each other. A graphical representation of the flow of implication in the heuristics of an individual expert is used to create a connection matrix for that expert.

Each expert is given a credibility weight and these weights are used to influence the decision processes of the fuzzy system. The initial weight for an expert is a measure of the experience of that expert. All weights are evaluated over time and the basis for evaluation is the success or failure of decisions made by the fuzzy system.

When an expert's rules are consistently successful, then that expert's credibility weight is increased by a small increment. On the other hand, when the expert's rules are consistently unsuccessful, then the expert's weighting is reduced. The size of the change to the credibility weight depends on two factors: the current credibility weight and whether the previous changes to this weighting were positive or negative. There is in-built resistance to changes in the credibility weights of experts who are very experienced and expert's with very little experience. The largest changes occur to

weightings between these two extremes.

In the case where a very experienced expert uses an ineffective or incorrect rule, this rule can be hidden or masked. Then, if this rule becomes effective or useful again, the rule is reintroduced to the reasoning processes of the fuzzy system. These techniques are tested using a fuzzy system for X-ray rocking curve analysis. This fuzzy system is presented in detail in Chapter 7.

4.2 Connection matrices and credibility weights.

For a particular application domain, the heuristics of an individual expert can be described using fuzzy knowledge networks (Kosko 1987). A fuzzy knowledge network is the graphical representation of the flow of implication in a set of heuristics derived from an individual expert through knowledge elicitation. Each premise and consequence is given a unique name and the heuristics of the individual expert are matched against the uniquely named rule components. Matching occurs through the use of a connection matrix. Connection matrices are described in terms of weights for and weights against inferences. These weights are in {-1, 0, 1}, where 1 is a positive implication; -1 is a negative implication; and 0 is assigned where no implication occurs.

4.2.1 Connection matrices.

A connection matrix for an individual expert is built from the heuristics used by that expert. The axes of the matrix are the unique names given to the premises and the consequences of all the rules used in the fuzzy system. Whenever the individual uses a particular rule, the premise and the consequence are linked in the matrix by the

value 1. When the individual disagrees with a particular rule, the premise and consequence are linked by the value -1. The value 0 means there is no connection between the premise and consequence for this expert. To illustrate this, five rules are taken from a fuzzy system for X-ray rocking curve analysis (Partridge and Tjahjadi 1994a). These rules are shown in Figure 4.1.

```
[A]  IF        type of structure is single layer = TRUE
       AND     peak splitting is zero = FALSE
     THEN      can calculate experimental mismatch = EXTREME
[B]  IF        type of structure is single layer = TRUE
       AND     substrate peak broadening = VERY
     THEN      bending of substrate = VERY
[C]  IF        type of structure is single layer = TRUE
       AND     peak separation is low = TRUE
       AND     layer asymmetry in peak = VERY
       AND     layer wedge shaped peak = VERY
     THEN      grading of the layer = VERY
[D]  IF        type of structure is single layer = TRUE
       AND     peak separation is low = TRUE
       AND     layer asymmetry in peak = EXTREME
       AND     layer wedge shaped peak = NONE
     THEN      grading of the layer = FAIRLY
[E]  IF        type of structure is single layer = TRUE
       AND     intensity of layer peak = EXTREME
     THEN      layer is thick = VERY
```

Figure 4.1 Five fuzzy rules for X-ray rocking curve analysis.

These rules use fuzzy variables and Boolean variables in the premises. The consequences use a single fuzzy variable. The five rules describe deductions based on values obtained for substrate and layer peaks. Four of the rules [A, B, C, E] are used by expert BKT (Professor B. K. Tanner of Durham University). Another expert DKB (Professor D. K. Bowen of Warwick University) uses just three of the rules, [A, C, D]. The experts NL (Dr. N. Loxley of Bede Scientific Instruments Ltd.) and CRT (Dr. C. R. Thomas of Warwick University) both use rules [A, E]. The connection matrices for each of the four experts is shown in Figure 4.2.

BKT

	AP	AC	BP	BC	CP	CC	DP	DC	EP	EC
AP	0	1	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0
BP	0	0	0	1	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	0	0	0
CP	0	0	0	0	0	1	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0
DP	0	0	0	0	0	0	0	0	0	0
DC	0	0	0	0	0	0	0	0	0	0
EP	0	0	0	0	0	0	0	0	0	1
EC	0	0	0	0	0	0	0	0	0	0

DKB

	AP	AC	BP	BC	CP	CC	DP	DC	EP	EC
AP	0	1	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0
BP	0	0	0	0	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	0	0	0
CP	0	0	0	0	0	1	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0
DP	0	0	0	0	0	0	0	1	0	0
DC	0	0	0	0	0	0	0	0	0	0
EP	0	0	0	0	0	0	0	0	0	0
EC	0	0	0	0	0	0	0	0	0	0

NL

	AP	AC	BP	BC	CP	CC	DP	DC	EP	EC
AP	0	1	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0
BP	0	0	0	0	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	0	0	0
CP	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0
DP	0	0	0	0	0	0	0	0	0	0
DC	0	0	0	0	0	0	0	0	0	0
EP	0	0	0	0	0	0	0	0	0	1
EC	0	0	0	0	0	0	0	0	0	0

CRT

	AP	AC	BP	BC	CP	CC	DP	DC	EP	EC
AP	0	1	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0
BP	0	0	0	0	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	0	0	0
CP	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0
DP	0	0	0	0	0	0	0	0	0	0
DC	0	0	0	0	0	0	0	0	0	0
EP	0	0	0	0	0	0	0	0	0	1
EC	0	0	0	0	0	0	0	0	0	0

Figure 4.2 Connection matrices for experts BKT, DKB, NL and CRT, where the leftmost column of the matrix is the premises and the horizontal row the consequence.

Each rule is split into its component premise and consequence, e.g. Rule [B] (in Figure 4.1) has the premise BP and consequence BC. The premises of all rules appear among the consequences, and vice versa, so that the consequence of one rule can appear as the premise of another. This feature is useful in many applications.

C_i implies C_j if the intersection of the row i and the column j of the connection matrix C is greater than 0. C_i does not imply C_j if the intersection is less than 0. C_i does not affect C_j if the intersection is equal to 0. In each of these cases, C_i is a premise and C_j is a consequence. The matrix isomorphism with the individual expert's heuristics creates a graphical representation that is easily combined with the heuristics of other experts. Different sets of knowledge used by different experts need not be equivalent.

4.2.2 Credibility weights and the combined connection matrix.

Each expert is given a credibility weight in the range [0, 1]. This is a measure of the experience of the expert. The individual expert's connection matrix is scalar multiplied by this credibility weight and all the connection matrices are added together using pointwise matrix addition. The resulting matrix describes the combined weighted heuristics of all the experts (Kosko 1987).

By assigning a weighting of 0.9, say, to BKT and 0.7 to DKB, and 0.6 to both NL and CRT, a combined matrix for these four experts is constructed using

$$0.9 \times \text{BKT} + 0.7 \times \text{DKB} + 0.6 \times \text{NL} + 0.6 \times \text{CRT}$$

Figure 4.3 shows the combined matrix for these four experts.

	AP	AC	BP	BC	CP	CC	DP	DC	EP	EC
AP	0	2.8	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0	0
BP	0	0	0	0.9	0	0	0	0	0	0
BC	0	0	0	0	0	0	0	0	0	0
CP	0	0	0	0	0	1.6	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	0
DP	0	0	0	0	0	0	0	0.7	0	0
DC	0	0	0	0	0	0	0	0	0	0
EP	0	0	0	0	0	0	0	0	0	2.1
EC	0	0	0	0	0	0	0	0	0	0

Figure 4.3 Combined connection matrix for experts BKT, DKB, NL, and CRT.

The generalised equation for n experts is

$$\sum_{i=1}^{n} w_i \times A_i$$

where w_i is the weighting and A_i the connection matrix for expert i.

Using this technique, a fuzzy system makes deductions on the basis of those rules with the highest weightings in the combined connection matrix. The rows of the matrix are read and whenever a value greater than zero occurs in a row, the premise for that row is stored. Next, the consequence with the highest weighting in the row is stored. When a row contains only zero values, and/or values less than zero, no premise or consequence is stored for that row. Once all the rows in the matrix are read, then the stored set of fuzzy rules are fired concurrently.

In the case where the conclusion of one rule acts as the premise of another, then all the rules cannot be fired concurrently. In this case, rules are represented in a series of layers. Rules with the highest precedence are at Layer 1. Layers of lesser precedence are at higher numbered layers. Thus, each rule has a layer number associated with it and rules are fired layer by layer. The rules in a single layer are all fired concurrently. The problem of inhomogeneity of rules belonging to a single expert can be dealt with in two ways: by masking ineffective rules (see Section 4.4);

and by fine-tuning and changing rules to deal with new facts (see Chapter 5).

4.2.3 Main advantages and disadvantages of this technique.

This technique can be used to combine the knowledge of numerous experts in a fuzzy system. The method avoids the problem of dealing with different levels of belief by incorporating a credibility measure based on experience. When the number of experts is large, then deviant heuristics become marginalised by the sheer weight of opinion towards the norm.

The obvious problem with this is that the prejudices of a community of experts will become embedded in the inference structures of the fuzzy system. A further problem is that this method will focus attention on those rules which are effective in dealing with the widest range of inputs and outputs. That is, the fuzzy system will use a set of rules that is the most effective in general. In a changing application domain, the method needs to be more specific than this. It must instead focus attention on those rules that deal most effectively with the latest, or most recent, inputs and outputs. To solve both these problems, the credibility weights of experts are evaluated and incrementally changed over time. The basis for evaluation is successful or unsuccessful decisions made by the fuzzy system.

4.3 Continuous evaluation of credibility weights.

The credibility weight of an expert is altered when an expert has consistently successful rules. In this instance, the credibility weight can be raised by a small increment. Conversely, if the rule set of a given expert is consistently unsuccessful, the fuzzy system may reduce the weighting of that expert. Success is defined in terms

of the performance of a fuzzy system in actual decisions. The performance measure used is derived to suit the application (a performance measure for X-ray rocking curve analysis is presented in Chapter 7). Hence, the fuzzy system is not a static structure. It is data driven, in the sense that real-life data can change the connections within the fuzzy system (Partridge and Tjahjadi 1994a).

4.3.1 Monitoring the behaviour of each expert's ruleset.

Each time a successful decision is fed back into the fuzzy system, the control structures monitor those rules that produced this correct decision. If any of an expert's rules is contained among this successful set of rules, then binary true (successful) is recorded for this expert. When an unsuccessful decision is fed back, a false value is recorded for all those experts whose rules were used in the unsuccessful decision. By this method, a history of the behaviour of each expert's ruleset is recorded.

4.3.2 Increasing or decreasing the credibility weight of an expert.

A percentage confidence limit is set for the effectiveness of an expert's ruleset. This confidence limit is *suc%*. Whenever an expert's ruleset is more than *suc%* effective in a sample of N decisions, then the credibility weight of that expert is increased by a small increment. The sample size N is defined to suit the application area. Conversely, a lower confidence limit is set for the ineffectiveness of an expert's ruleset. This confidence limit is *uns%*. An expert whose ruleset is more than *uns%* ineffective in N decisions will have his/her credibility weight reduced. These values of *suc%* effective and *uns%* ineffective decisions are decided after repeated simulations for different values.

These conditions for changing credibility weights can be stated as:

(1) IF $T(E_i) \geq suc\%$ *in* N decisions

THEN $cred_i = cred_i + inc_i$

(2) IF $F(E_i) \geq uns\%$ *in* N decisions

THEN $cred_i = cred_i - dec_i$

where $T(E_i)$ is the percentage of successful decisions recorded for expert i, $F(E_i)$ is the percentage of unsuccessful decisions recorded for expert i, $cred_i$ is the current credibility weight, inc_i is the increment and dec_i is the decrement.

4.3.3 Calculating the size of the increment/decrement to a credibility weight.

The size of the increment (or decrement) depends on two things: the current actual credibility weight and whether the previous increments to this weight were positive or negative. These constraints are defined by the following:

$$\begin{aligned} inc_i &\equiv (0.5 - |0.5 - W_i|) * f(p,q) * 1/\kappa \\ dec_i &\equiv (0.5 - |0.5 - W_i|) * f(p^{-1},q^{-1}) * 1/\kappa \end{aligned} \qquad (4.1)$$

where

$$f(p,q) = \frac{\min(p + 1, q + 1)}{\max(p + 1, q + 1)}$$

In equation (4.1), inc_i is an increment to the current credibility weight W_i of the expert i. dec_i is a decrement. Both values are defined by using the modulus operator to calculate the distance of the current weighting from the average (0.5). The closer the weighting is to the average, the greater the value of each increment. Hence, there is an in-built resistance to change when the expert is either very experienced or very inexperienced. It is difficult to decrease the credibility weight of an experienced expert

and to increase the weighting of an inexperienced one. p is the ratio of positive increments to negative decrements for this expert. These are actual increments and decrements implemented on the expert's credibility weight.

Each time an expert's heuristics produce *suc%* or more correct decisions in a sample a value is stored. When *uns%* incorrect decisions occur in a sample a value is also stored. Correct decisions are recorded as positive and incorrect decisions are recorded as negative. All these values are stored in a stack. q is calculated by removing the values one by one from the stack; each time a value is removed, a constant ρ is decremented by a small amount ϵ; then the value is multiplied by ρ; positive values are added to one variable, negative values are recorded in a second variable; q is the modulus of the ratio of these two values. In calculating q, greater weight is given to the values from the most recent samples. This is because the most recent decisions are also the most relevant to a changing application domain.

The function $f(p,q)$ is a correction factor that monitors any discrepancy between the proportion of correct (and incorrect) decisions and the number of increments actually implemented. The value of $f(p,q)$ should normally be very close to 1. When the changes made to an expert's credibility weight are erratic, and change quickly from increments to decrements and back again, then the size of the changes is reduced. Thus, those experts whose rules are consistently good will have larger increments implemented on their credibility weights. Experts with consistently bad decisions will have larger decrements on their weights. After testing for sensitivity (see Section 4.3.5), it was found that p and q closely follow each other. Sometimes one value is greater than the other and sometimes vice versa. Therefore, the max and min functions are used in order to force the value of f to be always in the interval

[0,1]. κ is a constant defined by experts in the field of application and defines the rate at which a credibility weight is allowed to increase or decrease.

4.3.4 Discussion of the behaviour of this technique.

If an expert has a low weighting and this expert has never had a positive increment, then there will be resistance by the fuzzy system to increasing that expert's weighting. But if the expert has had previous positive increments, then there will be less resistance to increasing the credibility weight for this expert. The more increments this expert receives, the larger those increments will be. That is, until the expert achieves a credibility weight greater than 0.5. This is because any increment (or decrement) to an expert's credibility weight depends on the actual value of that weighting. Thus, a positive increment added to an expert with a very high weighting will be asymptotic to the value 1. The same occurs for experts with very low weightings. Their weightings approach, but never reach, zero. It follows from this that experts with credibility weights around 0.5 have the heaviest increments and decrements to their weights. Hence, there is resistance to change at both the bottom and the top of the spectrum. An expert who is inexperienced will have difficulty increasing his/her weighting, and experts who are very experienced will encounter resistance to any reductions in their weightings.

4.3.5 Testing the behaviour of this technique.

This technique has been implemented in the fuzzy system for X-ray rocking curve analysis. In the case of X-ray rocking curve analysis, *suc%* is set at 75% effective and *uns%* is set at 20% ineffective. These figures were decided upon after

repeated simulations for different values using 100 sample decisions. When analysis is changed from batches of one type of structure to another, then the behaviour of the fuzzy system can change significantly. A small sample size allows suitably swift changes to be made to credibility weights. For this reason, experts agreed that the sample size $N = 20$. In calculating q, $\rho \equiv 1$ and $\epsilon = \rho/\kappa$. $\kappa = 20$.

A batch of 100 single-layer experimental rocking curves, of substrate GaAs and layer AlGaAs, were run through the fuzzy system and the behaviour of credibility weights and rules was monitored. These were used to test the sensitivity of the p and q values in Equation (4.1). In the case of changes made to the credibility weight of DKB, these values are graphed in Figure 4.4.

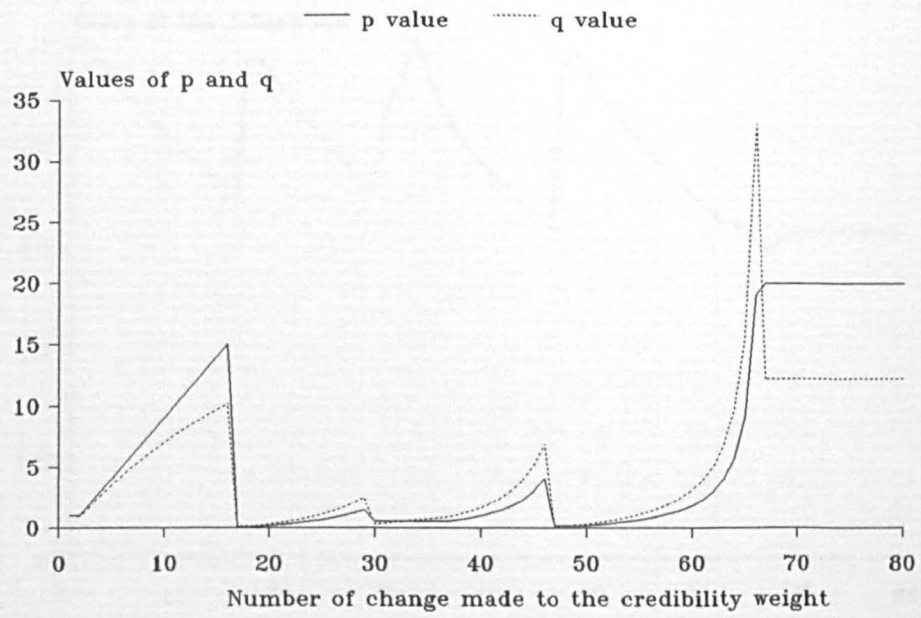

Figure 4.4 Comparison of p and q values for DKB in 80 changes to the credibility weight. One hundred sample decisions were used to test the sensitivity of the p and q values used in the f function.

It is found that the p and q values closely follow each other, with sometimes one value greater than the other and sometimes vice versa. Hence, the max and min functions are used in order to force the value of f to be always in the interval [0, 1]. It is also found that when the changes made to the credibility weight are consistently positive or consistently negative, then the value of p closely matches that of q. But when the changes shift from positive to negative increments, or vice versa, then these values diverge from each other. In this latter case, the effect on the function f is that its value is reduced from 1 and the size of the increment or decrement to the credibility weight is also reduced. This effect is illustrated in Figure 4.5.

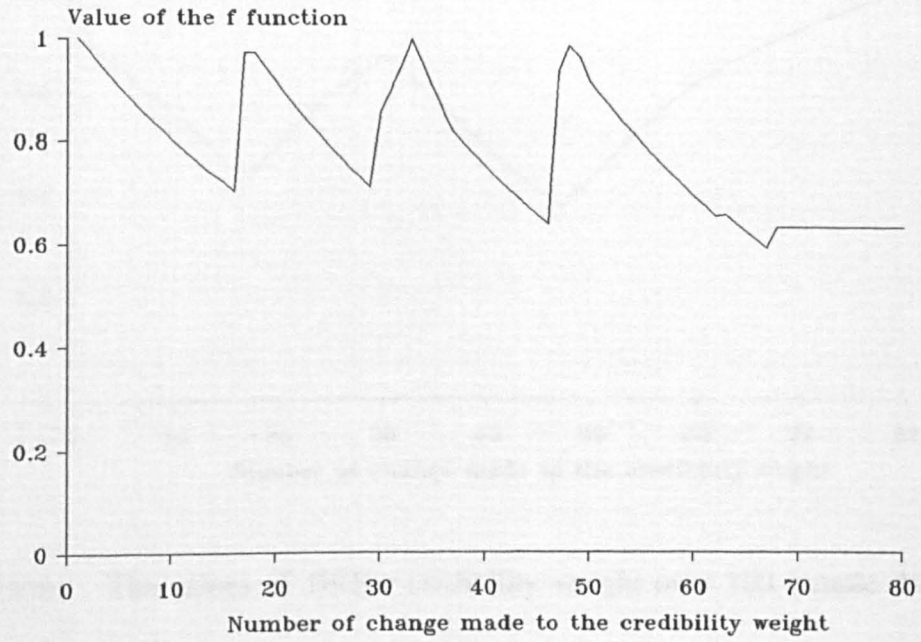

Figure 4.5 Changes in the value of function f for DKB in 80 changes to the credibility weight. The f function is used to calculate the size of the increment/decrement made to the credibility weight of the expert.

In order for an increment to occur in a credibility weight, 19 out of 20 decisions must be positive. Each new decision is added to the sample and the 20th previous decision is removed. If there are still 19 positive decisions, then a further increment is implemented. This will occur for as long as the 20 sample decisions contain 19 or more positive ones. In this manner, the changes in a credibility weight are accelerated so that the system quickly converges on its optimal set of credibility weights. A graph of the credibility weights for DKB, in 100 sample decisions, is shown in Figure 4.6.

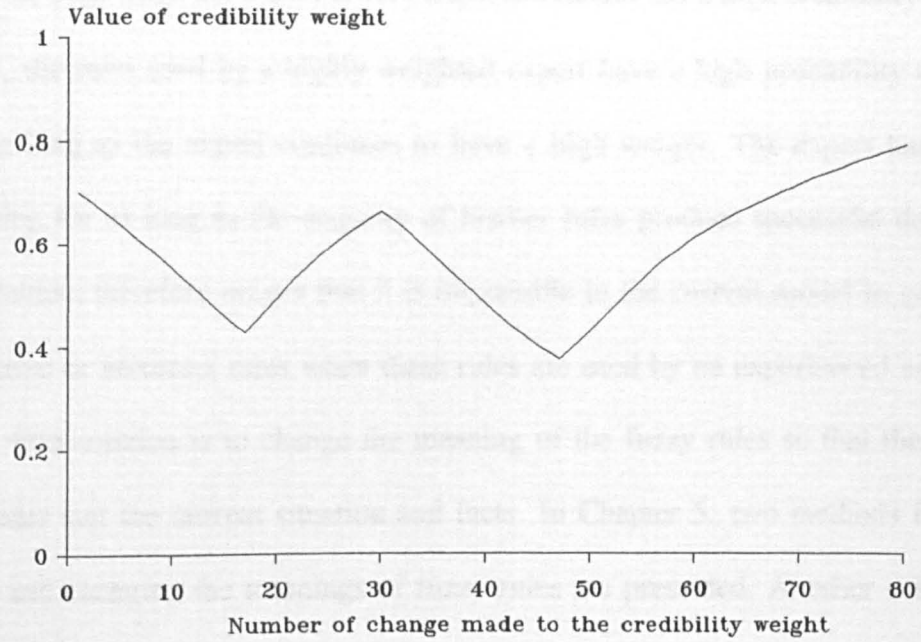

Figure 4.6 The values of DKB's credibility weight over 100 sample decisions. Eighty changes were made to the credibility weight in the 100 sample decisions. The lowest value of the credibility weight was 0.38 and the highest was 0.79.

Two sets of unsuccessful decisions adversely effect the credibility weight of DKB. These occur at decisions 15-19 and decisions 47-50. A number of 'difficult' rocking

curves were artificially included in the sample data at these points and affected the credibility weight by first reducing it to 0.43 and later reducing it to 0.38. In between, the credibility weight rose to 0.66 before later rising to a weighting of 0.8. These increments were incurred after rules used by DKB produced a series of successful decisions. The test data for these experiments is in Appendix A.

4.4 The masking of ineffective rules.

Sometimes a single rule produces incorrect decisions in a fuzzy system. This can occur even when the expert is very experienced and has a high credibility weight. In fact, the rules used by a highly weighted expert have a high probability of being used as long as the expert continues to have a high weight. The expert has a high weighting for as long as the majority of his/her rules produce successful decisions. The problem therefore occurs that it is impossible in the current model to get rid of ineffective or incorrect rules when these rules are used by an experienced expert.

One solution is to change the meaning of the fuzzy rules so that these rules will better suit the current situation and facts. In Chapter 5, two methods for fine-tuning and changing the meanings of fuzzy rules are presented. Another solution is to hide, or mask, the offending rules. This second solution is useful when a rule is so inappropriate that no fine-tuning will change the rule into an appropriate one. This solution uses two functions, called Unassert and Assert. These functions are derived from similar functions used in CYC (Guha and Lenat 1991).

4.4.1 The Unassert/Assert functions.

The masking functions work as follows: Unassert causes the fuzzy system to

mask or hide a rule; Assert causes the fuzzy system to reintroduce a rule already masked or hidden. A threshold is set in the fuzzy system which defines a certain proportion of unsuccessful decisions that are allowed for any rule. When a rule exceeds this threshold, this rule is masked by the Unassert function and it is removed from the reasoning processes of the fuzzy system. The rule is not "forgotten". In fact, a second threshold is set by the fuzzy system and all Unasserted rules are tested against this second threshold. A hidden rule is unmasked by the Assert function when that rule concurs with a certain proportion of successful decisions made by the fuzzy system. A hidden rule that is reasserted functions in exactly the same manner as before it was masked by the Unassert function.

CHAPTER 5. Fine-tuning and changing fuzzy rules to optimise the performance of the fuzzy system

5.1 Introduction.

In this chapter, two methods for fine-tuning fuzzy rules are presented. The purpose of this fine-tuning is to alter the meaning of fuzzy rules and thereby to customise the fuzzy system to deal with new situations and facts. The altered rules will describe new relationships between input and output data. These new relationships will be more accurate in dealing with the current situation presented to the fuzzy system. In a changing application domain, it is impossible to prescribe for all situations and facts in a fixed set of rules. Instead, the fuzzy system must track the alterations in the application domain and incrementally alter the meaning of rules so that these rules reflect the most recent situations and facts. The effectiveness of this fine-tuning is measured in terms of the performance of the fuzzy system.

The first method of fine-tuning uses a simplified version of fuzzy rules (Partridge and Tjahjadi 1994a). These rules have discrete values in their premises and these discrete values are incrementally changed. A sample of recent input values that produced good decisions by the fuzzy system is used to alter a rule. This information is also used to calculate the size of the change to the discrete value in the premise of the rule. This method is tested using a fuzzy system for X-ray rocking curve analysis (this system is presented in Chapter 7).

Simplified fuzzy rules restrict the effectiveness of the fuzzy system by restricting the deductive power of the underlying fuzzy logic. This deductive power is characterised by the use of linguistic values rather than discrete values in the premises of fuzzy rules. These linguistic values are matched against a set of overlapping triangles. Therefore, a second method is presented that fine-tunes and changes the triangular membership functions of fuzzy premise variables (Partridge and

Tjahjadi 1994b). A sample of recent inputs and outputs to the fuzzy system are used to alter the triangles. Values for successful decisions are compared with values for unsuccessful ones, and this comparison is used to both alter the width and move the triangles. Thus, the performance of the fuzzy system is optimised by changing the meaning of the fuzzy rules. Meaning here is defined as the relationship between the linguistic labels and the overlapping triangles. Optimisation is relative to the most recent situations and facts presented to the fuzzy system. This method is also tested using the fuzzy system for X-ray rocking curve analysis.

5.2 Fine-tuning and changing simplified fuzzy rules to deal with new facts.

This section presents a method for fine-tuning and changing simplified fuzzy rules. An example of a simplified rule is shown in Figure 5.1. This is a fuzzy rule for X-ray rocking curve analysis (Partridge and Tjahjadi 1994b).

```
IF          amount of interference fringes > 0.5
      AND   visibility of interference fringes > 0.4
THEN        layer is thick = FAIRLY
```

Figure 5.1 Rule 13: Simplified fuzzy rule for a single layer structure in X-ray rocking curve analysis.

In this example, two premises form the antecedent of the rule: `amount of interference fringes > 0.5`; and `visibility of interference fringes > 0.4`. These premises use discrete values and simple inequalities. The consequence of the rule, `layer is thick = FAIRLY`, uses a single fuzzy variable and a linguistic label taken from a term set. The term set is matched against a series of triangular

membership functions.

In order to fire this rule, input values are fuzzified and compared to the rule premises. Both premises must be satisfied completely before the rule is fired. In this case, the truth value of both premises is equal to 1. When a premise is not completely satisfied, then the truth value is zero. The simplified fuzzy rules are fired concurrently and the output value is calculated by defuzzification. For simplified fuzzy rules the additive output sets will not contain partial memberships.

5.2.1 Incremental changes to the premises of rules.

The method for fine-tuning and changing simplified fuzzy rules is: to present input data to the fuzzy ruleset; record the transactions for each rule; and incrementally adapt the existing rules to optimise their effects. Equation (5.1) is used to define incremental changes in the fuzzy measures of a rule. It is written in the form of two meta-rules:

(i) IF $\mu_A > \mu_R + 0.1$ in 90% of successful decisions

THEN $inc(\mu_R) = (0.5 - |0.5 - \mu_R|) * g(p, \kappa)$ (5.1)

(ii) IF $\mu_A < \mu_R - 0.1$ in 90% of successful decisions

THEN $dec(\mu_R) = (0.5 - |0.5 - \mu_R|) * g(p^{-1}, \kappa)$

WHERE $g(p, \kappa) = \frac{p + 1}{\kappa}$

In equation (5.1)

R is a rule stored in the fuzzy system;

A is a successful application of the rule R;

μ is a fuzzy measure in the interval [0, 1];

μ_R is the fuzzy measure used in the rule R;

μ_A is the actual fuzzy measure that produced a successful application of R;

p is the ratio of positive increments to negative decrements.

(These are actual increments and decrements implemented on the fuzzy measure of this rule); and

κ is a constant defined by experts in the application area. It defines the size of the increments and decrements.

A record of transactions is kept for each rule in a fuzzy system. This occurs even when the rule itself has not been applied. This seeming contradiction happens when, for example, `layer is thick = FAIRLY` but this was not indicated by the rule in Figure 5.1.

In the case where `visibility of interference fringes = 0.35` and `layer is thick = FAIRLY` the fuzzy system searches the rule-consequences and finds all those consequences that are satisfied in the current decision. In this instance, it is found that the layer is fairly thick even though `visibility of interference fringes < 0.4`. The value 0.35 is now recorded for the transactions of this rule and, if this feature occurs in 90% of decisions where the layer is fairly thick, then part (ii) of Equation (5.1) implements a decrement to the fuzzy measure (0.4) of the rule.

5.2.2 Testing the behaviour of this technique.

A batch of 100 single-layer experimental rocking curves, of substrate GaAs and layer AlGaAs, was run through a fuzzy system for X-ray rocking curve analysis. This fuzzy system used 83 simplified fuzzy rules. A change was implemented on one fuzzy variable in the fuzzy system. This variable occurs in Rule 13, shown in Figure

5.1. The change was implemented using part (i) of Equation (5.1).

In the case of a layer of AlGaAs on a GaAs substrate, the visibility of interference fringes is very high, so that each time Rule 13 is fired, the actual value of the fuzzy variable `visibility of interference fringes` is significantly higher than the value stored in the rule. The history of values for the fuzzy variable, `visibility of interference fringes`, used in Rule 13 is shown in Figure 5.2.

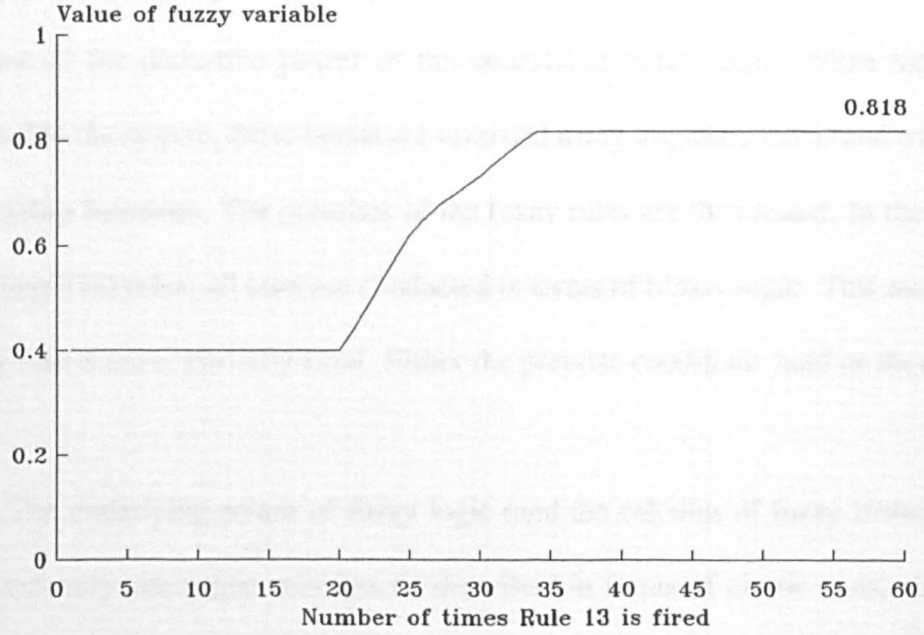

Figure 5.2 History of the values for the fuzzy variable `visibility of interference fringes`, used in Rule 13. This rule was fired 60 times in 100 sample decisions made by the fuzzy system.

After 28 of the 100 decisions, Rule 13 was fired 20 times. In 19 cases, the value of `visibility of interference fringes` was significantly higher than 0.4 and, so, the value in the rule was raised to 0.44. The experimental value continued to remain higher than that of the rule and by decision 40, the rule value had increased to 0.818.

At this point the rule reached equilibrium and the experimental value was no longer higher than that of the rule. The rule stabilized at this point. This behaviour demonstrates that a fuzzy rule may change its parameters in a meaningful way in as little as 40 decisions.

5.2.3 Limitations of simplified fuzzy rules.

Simplified fuzzy rules restrict the effectiveness of the fuzzy system because they restrict the deductive power of the underlying fuzzy logic. When inputs are presented to the system, these inputs are fuzzified using linguistic labels and triangular membership functions. The premises of the fuzzy rules are then tested. In the case of these simplified rules, all tests are conducted in terms of binary logic. This means that a fuzzy rule is never partially fired. Either the premise conditions hold or they do not hold.

The underlying power of fuzzy logic (and the calculus of fuzzy if/then rules) is that not only can vague concepts be described in terms of elastic constraints, but that these elastic constraints can be propogated using vague premises and vague consequences in fuzzy rules. This means that the premises of the fuzzy rules can be partially fired and more than one rule involving a consequence can be fired in any single decision made by the fuzzy system.

Kosko's Fuzzy Approximation Theorem (Kosko 1992b) proves that fuzzy concepts and fuzzy rules can be used to estimate the relationships between the inputs and outputs to the fuzzy system. These inputs and outputs are discrete and the fuzzy system that models these relationships does so precisely. Hence, an effective fuzzy system models not just the vague concepts, but also models vague relationships

defined as fuzzy rules. In this case, vague premises can be partially fired in order to infer vague consequences. These consequences are then defuzzified to calculate discrete output values.

For these reasons, the above technique is extended to deal with vague premises that can be partially fired. These fuzzy variables use triangular membership functions that are attached to a term set of linguistic labels (see Section 2.2). In order to fine-tune these fuzzy premise variables, the triangular membership functions must be incrementally changed. This alters the meaning of the linguistic label attached to the membership function. This technique is presented in detail in the next section.

5.3 Fine-tuning the membership functions of fuzzy premises.

Fine-tuning and changing the triangular membership functions of fuzzy premises alters the meaning of these premises (Partridge and Tjahjadi 1994b). The purpose of these changes is to customise the fuzzy rule to suit the most recent inputs and outputs to the fuzzy system. Inputs and outputs that produce incorrect or inaccurate decisions are used to alter the shape and size of particular triangles. These negative values are compared with positive values, and the clustering of the negative values determines the amount of change made to the triangle.

5.3.1 Tally of negative decisions involving a fuzzy rule.

A tally of negative decisions is kept for every rule stored in the fuzzy rulebase. There are as many tallies as there are rules and each tally is a sample of the most recent decisions involving a fuzzy rule. The definition of success and the size of the sample depend on the application. Whenever the tally of negative decisions for a

particular fuzzy rule rises above a threshold, then the fine-tuning algorithm is invoked.

5.3.2 Choosing which fuzzy premise to fine-tune.

The algorithm first identifies which fuzzy premise in the rule has the lowest membership value for the current decision. This lowest value is the support that the premises of the rule gave to the consequent of that rule. It is the particular value that fired the rule and hence produced the most recent negative decision. The fuzzy variable that produced this lowest value is the one that is fine-tuned.

5.3.3 Checking the distribution of negative values against positive ones.

Two sets of histories are used to fine-tune this fuzzy variable. These histories record a sample of recent values taken by the fuzzy variable and used in the premise of the rule. When a value was used in a positive decision, this value is stored in the positive history. A value used in a negative decision is stored in the negative history. These histories are used to decide the amounts of fine-tuning of the triangular function. Whenever the values in the negative history are clustered near the top of the triangular function, then the upper boundary of this triangle is reduced. When the values in the negative history are clustered near the bottom of the triangle, then the lower boundary of the triangle is increased. In order to quantify this level of clustering, the dispersion of the negative values is compared with the dispersion of the positive values.

A change is made to the triangular function only when the clustering of values in the negative history is significantly different to the clustering of values in the

positive history. In order to quantify this dispersion, the maximum value, the minimum value and the median value are taken from the positive history. Then, two tests are performed on each value in the negative history:

$$\text{TEST 1 :} \quad \text{IF } Neg\text{-}His_j > pos\text{-}max_j - \left(\frac{pos\text{-}max_j - median_j}{5} \right)$$

$$\text{TEST 2 :} \quad \text{IF } Neg\text{-}His_j < pos\text{-}min_j - \left(\frac{median_j - pos\text{-}min_j}{5} \right)$$

where $Neg\text{-}His_j$ is a value in the negative history, $pos\text{-}max_j$ is the maximum value in the positive history, $pos\text{-}min_j$ is the minimum value in the positive history and $median_j$ is the median value.

In TEST 1, a threshold is set just below the maximum value in the positive history. The level below this maximum is set at 20% of the dispersal of the top 50% of positive values. This allows for a small overlap between positive and negative values. Each value in the negative history is tested against this threshold. In TEST 2, a similar threshold is set just above the minimum value in the positive history. All negative values are then tested against this second threshold.

5.3.4 Incremental changes to the primary and secondary triangles.

When TEST 1 is satisfied to a significant degree, then the upper boundary of the triangular function is reduced by the following amount:

$$Dec_j = (0.5 - |0.5 - upper\text{-}bound|) * \frac{range\text{-}neg_j}{10}$$

where Dec_j is the size of the decrement to the upper boundary, *upper-bound* is the current upper bound of the triangular function, and $range\text{-}neg_j$ is the range of values in the negative history. The size of the decrement depends on two factors: the distance

of the triangle from 0 and 1; and the range of values in the negative history. Triangles in the centre of the interval [0, 1] and with a large range of negative values will incur the largest changes to their bounds. The lower boundary of the triangle above the primary one is reduced by $\frac{1}{2} \times Dec_j$. The degree of significance to which the test is satisfied is a percentage confidence limit decided after repeated simulations for different values using test data from the application domain.

When TEST 2 is satisfied, then the lower boundary of the triangular function is increased. The size of the increase is:

$$Inc_j = (0.5 - |0.5 - \textit{lower-bound}|) * \frac{\textit{range-neg}_j}{10}$$

where Inc_j is the size of the increment and *lower-bound* is the current lower bound of the triangular function. The upper bound of the triangle below the current one is increased by $\frac{1}{2} \times Inc_j$.

It can be seen that the triangles that surround the altered triangle are also fine-tuned. But the changes to these triangles are less than to the primary one. When a secondary triangle changes too much, then it will produce negative decisions. It will then become a primary triangle and will itself become altered by the algorithm until it reaches equilibrium. Three constraints are placed on changes that can be made to triangular functions: the amount of overlap between triangular functions is kept to between 10% and 50% of the neighbouring space; the lowest triangle is not allowed to move away from 0 and the top triangle is not allowed to move down from 1; and the sum of the vertical points of the overlap is always less than one. The first two constraints avoid empty spaces appearing in the coverage of the interval [0, 1], while the third constraint avoids problems with calculating the added output.

5.3.5 Algorithm for fine-tuning the membership functions of fuzzy premises.

This algorithm can be stated more formally as:

Definitions:-

Let r_i be a fuzzy rule, x_j the fuzzy premise with the lowest membership function in the current decision and $tally_i$ is the tally of negative decisions for r_i.

Let T_0 be the triangular function that is currently being fine-tuned and changed, T_{+1} and T_{-1} are the triangular functions above and below T_0, respectively. Let the widths of these triangular functions be $w(T_0)$, $w(T_{+1})$ and $w(T_{-1})$.

Algorithm:-

CASE: $tally_i$ > threshold

CASE 1:

$\text{Neg-His}_j > pos\text{-}max_j - \left(\frac{pos\text{-}max_j - median_j}{5}\right)$ is true 80% of the time

AND upper-bound(T_0) ≠ 1.0

AND max(membership(T_0) + membership(T_{+1})) ≤ 1.0

THEN Reduce *upper-bound*(T_0) by

$$Dec_j = (0.5 - |0.5 - upper\text{-}bound(T_0)|) * \frac{range\text{-}neg_j}{10}$$

AND reduce *lower-bound*(T_{+1}) by ½ × Dec_j.

CASE 2:

$\text{Neg-His}_j < pos\text{-}min_j - \left(\frac{median_j - pos\text{-}min_j}{5}\right)$ is true 80% of the time

AND lower-bound(T_0) ≠ 0.0

AND max(membership(T_0) + membership(T_{+1})) ≤ 1.0

THEN Increase *lower-bound*(T_0) by

$$Inc_j = (0.5 - |0.5 - \textit{lower-bound}(T_0)|) * \frac{\textit{range-neg}_j}{10}$$

AND increase *upper-bound*(T_{-1}) by ½ × Inc_j.

AND

CASE A:

$$\textit{upper-bound}(T_0) < \textit{lower-bound}(T_{+1}) + \frac{w(T_{+1})}{10}$$

THEN Reset the upper bound to 10% overlap, i.e.

$$\textit{upper-bound}(T_0) = \textit{lower-bound}(T_{+1}) + \frac{w(T_{+1})}{10}$$

CASE B:

$$\textit{upper-bound}(T_0) > \textit{lower-bound}(T_{+1}) + \frac{w(T_{+1})}{2}$$

THEN Reset the upper bound to 50% overlap, i.e.

$$\textit{upper-bound}(T_0) = \textit{lower-bound}(T_{+1}) - \frac{w(T_{+1})}{2}$$

CASE C:

$$\textit{lower-bound}(T_0) > \textit{upper-bound}(T_{-1}) - \frac{w(T_{-1})}{10}$$

THEN Reset the lower bound to 10% overlap, i.e.

$$\textit{lower-bound}(T_0) = \textit{upper-bound}(T_{-1}) - \frac{w(T_{-1})}{10}$$

CASE D:

$$\textit{lower-bound}(T_0) < \textit{upper-bound}(T_{-1}) - \frac{w(T_{-1})}{2}$$

THEN Reset the lower bound to 50% overlap, i.e.

$$\textit{lower-bound}(T_0) = \textit{upper-bound}(T_{-1}) - \frac{w(T_{-1})}{2}$$

5.3.6 Justification of the fine-tuning algorithm.

By changing the membership functions of fuzzy premises, the relationship between triangular functions and the set of linguistic labels is changed. The values that can be taken by a linguistic variable are incrementally altered so that the performance of the fuzzy rule is optimized. That is, the values that produce negative decisions are identified and then the meanings of the linguistic variables are shifted away from these negative values towards those values that produced positive decisions. In this way, the performance of a fuzzy rule is optimized. Optimization is relative to the most recent situations and facts presented to the fuzzy system.

5.3.7 Testing the behaviour of this technique.

This algorithm has been implemented in a fuzzy system for X-ray rocking curve analysis. To test the algorithm, 100 single-layer experimental rocking curves, of substrate GaAs and layer AlGaAs, were run through the system. A sample size of 20 decisions was used to record the previous decisions for each fuzzy rule. Decisions were recorded as either positive or negative. 20 decisions were used because it was found that this sample size produced fine-tuning in a convenient period of time for this application. The samples of previous decisions were updated after each decision was made by the fuzzy system. Then, after each update, the lists of decisions were read. When the tally of negative decisions was greater than 80%, then the fine-tuning algorithm was used. In these tests, the membership functions of three fuzzy premises were changed during 100 decisions involving these rules. The changed rules were Rules 11, 21 and 30 out of 117 rules for single layer structures. Rule 30 is used to illustrate this fine-tuning. This rule is shown in Figure 5.3.

```
IF      type of structure is single layer = TRUE
   AND  layer asymmetry in peak = NONE
   AND  layer wedge shaped peak = SOME
   AND  peak splitting less than 150 = TRUE
THEN    grading of the layer = NONE
```

Figure 5.3 Rule 30 for single layer structures in fuzzy system for X-ray rocking curve analysis.

The membership functions for the fuzzy variable `layer asymmetry in peak` is shown in Figure 5.4. The term set for this variable is {NONE, SOME, FAIRLY, VERY, EXTREME}.

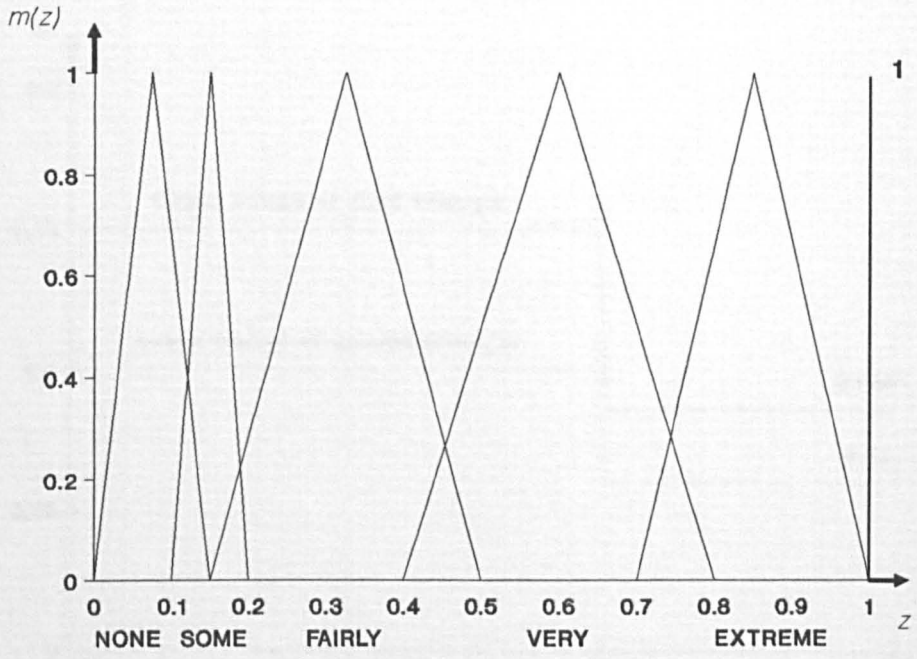

Figure 5.4 The triangular membership functions for the fuzzy premise variable `layer asymmetry in peak`.

The boundaries of the triangular function for `layer asymmetry in peak = NONE` were 0.0 and 0.15. The boundaries for the next triangular function, `layer asymmetry`

in peak = SOME were 0.1 and 0.2. In 100 decisions involving this rule, the inputs to the fuzzy premise layer asymmetry in peak = NONE were always either 0.05 or 0.1. This was because question and answer sessions with the user were used to provide the inputs to the fuzzy system. For Rule 30, all positive decisions were fired by the 0.05 value. In no cases did a value of 0.1 provide a positive decision. In 2 cases (i.e. 2⅔%) the value 0.05 produced negative decisions. After 65 decisions, the upper boundary of the first triangular function changed from 0.15 to 0.086. The lower boundary of the second triangle changed from 0.1 to 0.06. This change isolated the value 0.1 from the range of values that fired the rule and thereby optimised the performance of Rule 30. These changes are shown in Figure 5.5.

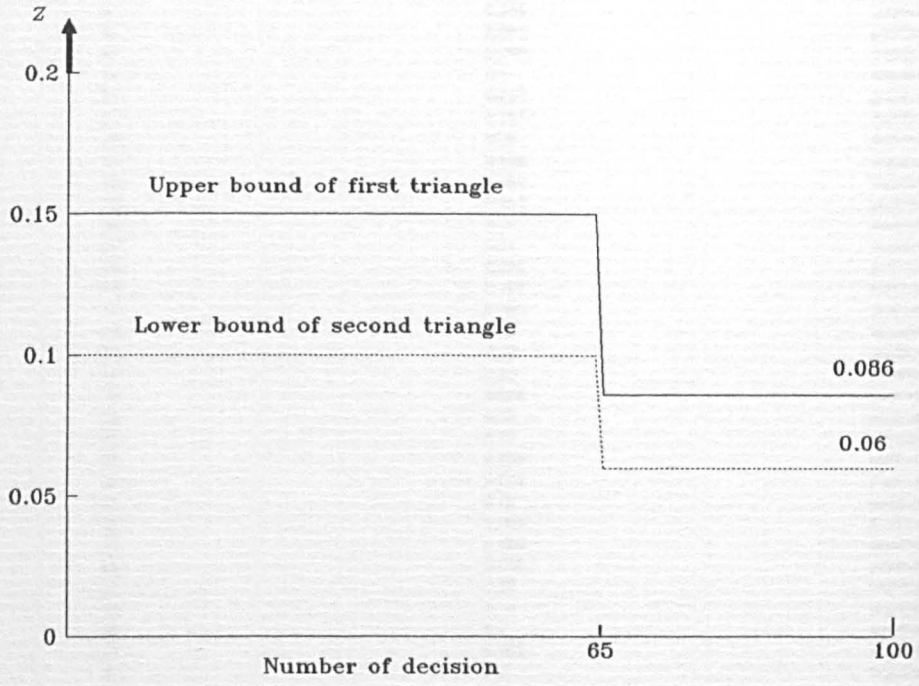

Figure 5.5 Changes made to the boundaries of the triangular functions for the fuzzy premises layer asymmetry in peak = NONE and layer asymmetry in peak = SOME.

The changes made to other fuzzy premises also changed the lowest triangular functions. The size of the changes were very similar to the changes made to Rule 30. This was because the user entered the amount of asymmetry in a peak using menus in question and answer sessions.

In the case where inputs are measured directly from the graph, a more accurate measure of asymmetry will produce a wider variation in input values. To investigate this, a set of simulated values was inputed to the fuzzy system. These values are shown in Table 5.1.

Number of Decision	Value of layer asymmetry	Value of Decision
1:	0.05	positive
2:	0.076	negative
3:	0.014	positive
4:	0.056	positive
5:	0.045	positive
6:	0.089	negative
7:	0.1	negative
8:	0.058	positive
9:	0.065	positive
10:	0.06	positive
11:	0.098	negative
12:	0.099	negative
13:	0.012	positive
14:	0.045	positive
15:	0.11	negative
16:	0.035	positive
17:	0.045	positive
18:	0.087	positive
19:	0.11	negative
20:	0.07	negative
21:	0.06	positive
22:	0.078	positive
23:	0.88	negative
24:	0.09	negative
25:	0.033	positive
26:	0.044	positive
27:	0.05	positive
28:	0.045	positive
29:	0.086	negative
30:	0.066	positive
31:	0.89	negative
32:	0.12	negative
33:	0.1	negative
34:	0.098	negative
35:	0.1	negative
36:	0.09	negative
37:	0.05	positive
38:	0.063	positive
39:	0.07	positive
40:	0.056	positive

Table 5.1. Set of simulated values for the fuzzy premise `layer asymmetry in peak = NONE`.

These simulated values produced a change to the boundaries of the triangular functions of Rule 30 after 36 decisions. These changes are shown in Figure 5.6.

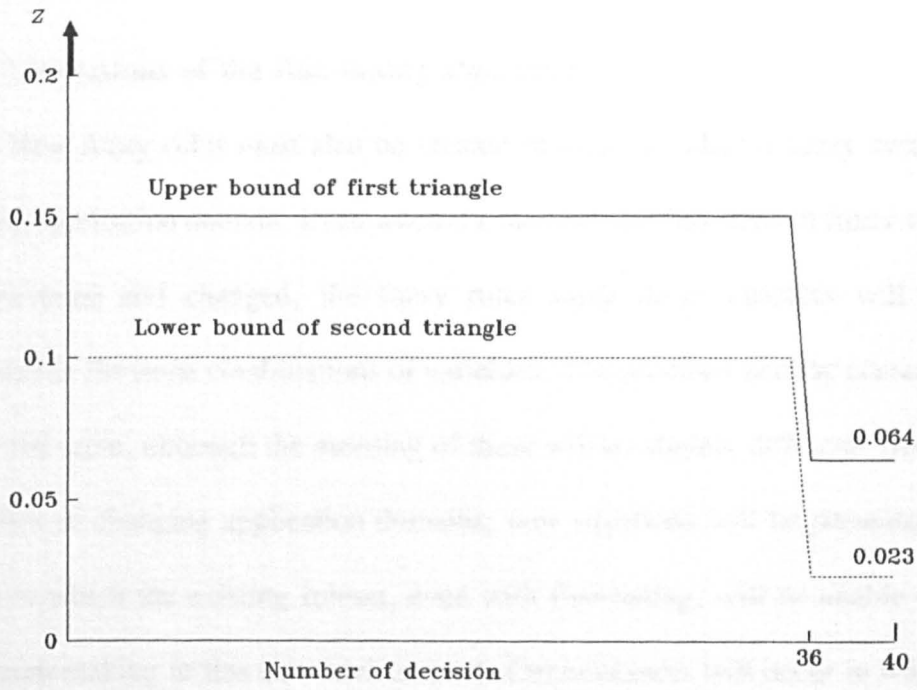

Figure 5.6 Changes made to the boundaries of the triangular functions for the fuzzy premises `layer asymmetry in peak = NONE` and `layer asymmetry in peak = SOME` using simulated input values.

In this case, the upper boundary of `layer asymmetry in peak = NONE` changed from 0.15 to 0.064, while the lower boundary of `layer asymmetry in peak = SOME` changed from 0.1 to 0.023. These changes isolated all the negative input values, but they also isolated 20% of the positive values too. So, the relative performance of Rule 30 was optimised but at a cost. The four positive values that were lost to Rule 30 move into the area of the triangular function for the fuzzy premise `layer asymmetry in peak = SOME`. Only when these values adversely affect a rule using this premise

will the boundary of layer asymmetry in peak = NONE change again to include these values.

5.3.8 Limitations of the fine-tuning algorithm.

New fuzzy rules must also be created in order to adapt a fuzzy system to a changing application domain. Even when the membership functions of fuzzy variables are fine-tuned and changed, the fuzzy rules using these variables will still be composed of the same combinations of variables. The premises and the consequences will be the same, although the meaning of these will be slightly different. Because of the nature of changing application domains, new situations will be presented to the system in which the existing ruleset, even with fine-tuning, will be unable to adapt to decision-making in this new environment. Circumstances will occur in which new combinations of variables will be needed in order to make correct decisions. Therefore, it is important to include mechanisms that will allow the fuzzy system to generate new fuzzy rules. These rules will describe new relationships between inputs and outputs to the fuzzy system, relationships that describe new situations and facts in the changing application domain. An inductive learning algorithm that creates new fuzzy rules is presented in Chapter 6.

CHAPTER 6. Rule generation by inductive learning from examples

6.1 Introduction.

This chapter presents an algorithm for generating and testing new fuzzy rules. The algorithm uses inductive learning from examples (Forsyth 1989, Arunkumar and Yagneshwar 1990). This algorithm is particularly useful when a fuzzy system is being applied in a changing application domain.

In changing application domains, new relationships can occur between inputs and outputs that have not been prescribed for in the fuzzy rulebase. Fine-tuning of existing rules will deal with many of these new relationships (see Chapter 5), but circumstances can occur where new combinations of input and output variables are necessary. By tracking recent input values and output values these new combinations can be detected. Inductive learning can be used to infer new rules that relate these input and output values. In inductive learning, the evidence for a new fuzzy rule is statistical in nature. It does not have a basis in formal logic.

An inductive learning algorithm creates an example set from a series of recent good decisions made by a fuzzy system. In a changing application domain, recent decisions are more typical of the current state of the domain and rules created from these values will be more effective in making new decisions. The example set is recorded cumulatively in two induction test matrices and then a set of potential fuzzy rules is generated using simple statistical tests. Each of these potential rules is then tested using fitness functions. These fitness functions are derived from six evaluation criteria for knowledge-based systems (Guida and Mauri 1993). These well-established criteria are: correctness; coverage; robustness; effectiveness; simplicity; and naturalness. Only those potential rules that satisfy these fitness functions are added to the fuzzy rulebase. This technique is tested using an example set of input and output

values from a fuzzy system for X-ray rocking curve analysis (this system is presented in detail in Chapter 7).

6.2 Inductive learning of new fuzzy rules.

Inductive learning from examples derives general rules or laws from a set of particular examples. In a changing application domain, the example set is a series of recent inputs and outputs to the fuzzy system. These inputs and outputs are taken from good decisions, that is decisions with a high success rate or low error rate in the application domain. This section presents an inductive learning algorithm that creates new fuzzy rules in a changing application domain.

6.2.1 The induction test matrices.

The inductive learning algorithm uses two matrices to create new fuzzy rules. These are called Positive Induction Test (PIT) matrix and Negative Induction Test (NIT) matrix. The first matrix records cumulative values for positive examples. These are values for the inputs and outputs of the fuzzy system when the system produces very good decisions. The quality of a decision is defined in terms of a success rate or an error rate for the particular application domain. The empty generalised PIT matrix is shown in Figure 6.1.

The PIT matrix has a column for each linguistic label (Pm_i) taken by each fuzzy variable used in the premises of rules. There is a row for each linguistic label (Cn_j) taken by each fuzzy consequent. There are two extra columns: one column *(C)* for the cumulative consequent values and one *(CT)* for a tally of the number of occurrences of this consequent in the example set. Thus, the dimensions of the PIT

	$P1_1$	$P1_2$	$P1_3$	$P1_4$	$P1_5$	$P2_1$	$P2_2$	$P2_3$	$P2_4$	$P2_5$	.....	Pm_1	Pm_2	Pm_3	Pm_4	Pm_5	C	CT
$C1_1$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C1_2$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C1_3$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C1_4$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C1_5$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C2_1$	0	0	0	0	0	0	0	0	0	0	.....	0	0	0	0	0	0	0
$C2_2$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C2_3$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C2_4$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
$C2_5$	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
.						.								.				
.						.								.				
.						.								.				
Cn_1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
Cn_2	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
Cn_3	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
Cn_4	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
Cn_5	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

Figure 6.1 The generalised PIT matrix where each fuzzy variable has 5 triangular membership functions. The matrix is empty.

matrix are: $(m \times i) + 2 \times (n \times j)$, where m is the number of fuzzy premise variables, n is the number of fuzzy consequences, i is the number of linguistic labels in the term set for fuzzy premise variables and j is the number of linguistic labels for fuzzy consequent variables. In fact, the values of i and j can be the same or they can differ for individual fuzzy variables.

For example, if there are 2 fuzzy premises taking 5 linguistic values each and there is 1 consequent also taking 5 linguistic values, then the dimensions of the PIT matrix is $(2 \times 5) + 2 \times (1 \times 5)$. The empty PIT matrix for these premises and consequent is shown in Figure 6.2. In this example, columns A1, ..., A5 will contain the cumulative totals for each of the premise values of fuzzy variable A, columns B1, ..., B5 will contain the totals for fuzzy variable B, column C will contain the cumulative total for the fuzzy consequent and column CT the tally of occurrences for

	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C	CT
C1	0	0	0	0	0	0	0	0	0	0	0	0
C2	0	0	0	0	0	0	0	0	0	0	0	0
C3	0	0	0	0	0	0	0	0	0	0	0	0
C4	0	0	0	0	0	0	0	0	0	0	0	0
C5	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6.2. An empty PIT matrix for 2 fuzzy premises A and B and 1 fuzzy consequent C. Each fuzzy variable has 5 triangular membership functions.

each consequent value. The rows C1, ..., C5 correspond to each consequent value of fuzzy variable C. The NIT matrix has exactly the same dimensions as the PIT matrix and records cumulative values for negative examples.

6.2.2 The example set used for the inductive learning.

The inductive learning algorithm uses an example set of recent instances of very good decisions in order to fill these matrices. These are decisions with a high success rate or a low error rate. Each example is taken in turn and a set of values are calculated and stored in the induction test matrices. First the output values for each of the consequents are used. The membership value for each triangular membership function is calculated for each fuzzy consequent. That is, each defuzzified output value is again fuzzified. The input values for each of the fuzzy premises are also fuzzified and a membership value is calculated for each membership function of each fuzzy premise. These sets of values are the basic data used to fill the PIT and NIT matrices.

6.2.3 Constructing the induction test matrices from the example set.

A positive threshold for consequent values is set very high (say, 0.95) and a

negative threshold is set very low (say, 0.125). Whenever the membership value for a consequent is above the positive threshold, the consequent value is added cumulatively to the PIT matrix at the appropriate position. All the premise values are added to the PIT matrix at the same row. Finally, the tally of instances of this consequent is incremented by 1. When the consequent value is below the negative threshold, these values are added to the NIT matrix. Values between the thresholds are ignored by the algorithm. Each example in the example set is used in turn, and the PIT and NIT matrices are gradually filled. The values in these matrices are then used to create new fuzzy rules.

6.2.4 Creating potential rules from the Positive Induction Test matrix.

In order to create these rules, each value taken by a fuzzy premise or consequent is uniquely indexed by a number or character. This indexing is similar to the indexing of rule elements used in genetic algorithms (Johnson and Feycock 1991). A minimum sample size is set (this depends on the application) and a number of potential rules are created from the PIT matrix. Each row in the PIT matrix is tested in turn for potential rules. The tally of occurrences of a consequent is tested to see if its value is above the minimum sample size:

TEST 1: $tc_j \geq min_{ss}$

where tc_j is the tally of occurrences of the consequent j and min_{ss} is the minimum sample size. Next, each cumulative premise value is tested to see if its value is above a minimum threshold defined in terms of the cumulative value of the consequent:

TEST 2: $fp_{ij} > pos * fc_j$

where fp_{ij} is the ith cumulative fuzzy premise value for the jth consequent, pos is the

positive threshold and fc_j is the cumulative value for the consequent j. Each premise that satisfies TEST 2 becomes the premise of a potential rule and the representation of its rule elements are stored. These elements are combined with other premises using the AND-connective. The representation of the fuzzy consequent is stored with these to form the complete rule. Rules created by this method use only the AND-connective. This is because each premise is cumulatively combined with other premises in forming the rule. Each potential rule is also checked against the existing ruleset and removed if it already exists.

6.3 Testing new rules using six fitness functions.

New fuzzy rules must be tested before they are incorporated into the fuzzy system. These tests must be both rigorous and based on the correct evaluation criteria. Fitness functions are mathematical formalisations of evaluation criteria for intelligent systems. A number of fitness functions have been implemented (see Section 2.4.3), but these systems have concentrated on just one or two criteria. A survey of 66 evaluation criteria for knowledge-based systems that were published between 1982 and 1990 was recently performed (by Guida and Maura 1993). Based on this survey, a precise definition of performance and quality was proposed and then a set of criteria was developed to support the practical implementation of this definition. Of these criteria, only six are applicable to individual fuzzy rules, the others (e.g. user-friendliness, hardware system etc.) are related to the overall intelligent system. These six criteria are: correctness; coverage; robustness; simplicity; effectiveness; and naturalness. Six new fitness functions are derived for each of these criteria and presented in this section.

6.3.1 Fitness function for correctness.

The correctness of a potential fuzzy rule is defined in terms of the evidence against that rule. This evidence is stored in the Negative Induction Test (NIT) matrix. If the negative evidence is below a threshold, then the potential rule satisfies the condition for correctness. Negative evidence is defined as a large amount of evidence for the premises of a rule given that there is no evidence for the rule consequent. The fitness function for correctness is given by:

$$cor = \frac{(Nfp_{ij} \div Ntc_j)}{(Pfc_j \div Ptc_j)} \quad \text{when } Ntc_j > 0$$

$$= 1 \quad \text{otherwise}$$

where Nfp_{ij} is the cumulative value of fuzzy premise i for consequent j in the NIT matrix, Ntc_j is the tally of negative occurrences of consequent j, Pfc_j is the cumulative value of consequent j in the PIT matrix and Ptc_j is the tally of positive occurrences of this consequent.

If the positive evidence for the premise i is high, then the value Nfp_{ij} is high. If the tally of occurrences of the consequent j is low, then the value of Ntc_j is low. In this case the calculated value of $(Nfp_{ij} \div Ntc_j)$ is high. Alternatively, if there is no evidence against the rule $i \rightarrow j$, then the value of $(Nfp_{ij} \div Ntc_j)$ is very low. The higher the value of $(Pfc_j \div Ptc_j)$ the more inductive evidence there is for the rule $i \rightarrow j$ in the PIT matrix. Therefore, if the negative evidence for a rule is high, then the *cor* function produces a high value. If there is little evidence for the rule, then the *cor* function produces a low value. The more positive evidence there is for the rule, the lower the value of the *cor* function. A threshold is set, and only when *cor* is less than this threshold does a potential rule satisfy the fitness function for correctness. The

value of this threshold depends on the application.

6.3.2 Fitness function for coverage.

The coverage of a potential rule is the number of rules that adequately covers a particular consequent. The fitness function for coverage tests the number of existing rules that contain a particular consequent against a predefined quota. This quota ensures that the fuzzy system is not filled up with a large quantity of rules that cover only a small number of consequents. If the number of rules is greater than the quota, then the potential rule is tested for robustness. This fitness function is given by:

$$cov = number(fc_j)$$

where *number* calculates the number of existing rules with consequent fc_j.

6.3.3 Fitness function for robustness.

The robustness of a potential rule is defined in terms of the average value of the consequent in the PIT matrix. The higher the average value of the consequent, the stronger the evidence is for the rule. If the average value is very high, then the evidence for the rule is very high and the rule itself is robust in terms of this statistical evidence. The fitness function for robustness is:

$$rob = \frac{Pfc_j}{Ptc_j}$$

where Pfc_j is the cumulative value of the consequent j in the PIT matrix and Ptc_j is the tally of positive occurrences of this consequent.

If the value of this fitness function is very high, i.e. 0.975, then the potential rule satisfies the test for robustness. This fitness function is used only when a potential

rule fails the test for coverage. This means that when there are lots of rules covering a particular consequent, a new rule is created only when the amount of evidence for this rule is very high.

6.3.4 Fitness function for effectiveness.

Effectiveness is defined in terms of low coverage for a fuzzy consequent. The basis for this is that, if a low number of rules cover a particular consequent then the fuzzy system is not effectively covering the full range of output values. Effectiveness in this context is defined in terms of overall effectiveness of the fuzzy rulebase. The fitness function for effectiveness is:

$$\begin{aligned} eff &= 1 \quad \text{when } low(cov) \\ &= 0 \quad \text{otherwise} \end{aligned}$$

where *low* defines low coverage in the current application. If there are very few rules covering a particular consequence, then the thresholds are artificially changed so that there is a higher chance of a new rule being created for this consequent than for others. These artificial changes are made to the thresholds used to define positive and negative instances for the induction test matrices.

6.3.5 Fitness function for simplicity.

The fuzzy system that uses simpler rules is more efficient, because it performs a smaller number of tests in firing its rules. If two rules infer the same consequence and both rules are equally successful in performing this task, the rule that uses a greater number of premise variables is actually testing for irrelevant information. Therefore, simplicity is defined in terms of the number of premises in a rule.

The fitness function for simplicity is:

$sim = 1$ when $premises(nr_{ij}) \subset premises(er_{ij})$

$= 0$ otherwise

where nr_{ij} is a new rule and er_{ij} is an existing rule, with premises i and consequent j, and *premises* is an index of the set of premises of a rule.When the consequent of a potential rule is the same as an existing rule in the fuzzy rulebase, the premises of the rules are compared. Whenever the premises of one rule is a subset of the premises of the other rule, then the rule with a lower number of premises is chosen. That is, the simpler rule satisfies the test for simplicity. When the more complex rule is in the existing fuzzy ruleset, then this rule is hidden from the reasoning processes of the fuzzy system. It is not completely removed from the rulebase. This is because this rule may be relevant to situations that were not included in the example set used for the inductive learning.

6.3.6 Fitness function for naturalness.

Naturalness is defined in terms of the user, who is used interactively to test potential rules. The fitness function for naturalness simply displays each potential rule and the user is asked to decide whether the rule is sensible. This formulation is derived from (Rada 1991), which argues that an appreciation of the relation between representation and meaning requires an immersement in the world and therefore the user should be used interactively to test predicates, rule variables etc. The fitness function for naturalness is:

$nat = 1$ when user accepts nr_{ij}

$= 0$ otherwise

6.4 Algorithm for generating and testing new fuzzy rules by inductive learning from examples.

This inductive learning algorithm creates new fuzzy rules from an example set of recent good decisions involving the fuzzy system. Only rules with statistical evidence for their premises and consequents are created. Then these new rules are tested using six fitness functions based on six evaluation criteria. Only those rules that test positive for these evaluation criteria are added to the fuzzy rulebase. Thus, new rules are created that deal with the most recent facts and situations in a changing application domain.

6.4.1 Formal representation of the algorithm.

This algorithm can be stated more formally as follows:

Definitions:-

E is the example set used to create new rules, $e \in E$ is an example taken from this example set; $min(nr_{ij})$ is the minimum number of new fuzzy rules to be created by the algorithm; *pos* is the positive threshold (say, 0.95) and *neg* is the negative threshold (say, 0.125).

fc_j is the value of fuzzy consequent j and fp_i is the value of fuzzy premise i in an example $e \in E$; Pfc_j is the cumulative value for consequent j in the PIT matrix; Nfc_j is the cumulative value in the NIT matrix; Pfp_{ij} and Nfp_{ij} are the cumulative values for fuzzy premises in the PIT and NIT matrices, respectively; Ptc_j and Ntc_j are the tallies of occurrences of fuzzy consequents in the PIT and NIT matrices, respectively.

nr_{ij} is a new fuzzy rule and er_{ij} is an existing fuzzy rule already in the fuzzy system.

Algorithm:-

CASE: number of new rules < $min(nr_{ij})$

CASE 1: $fc_j > pos$

THEN Add fc_j to the cumulative value Pfc_{ij}

AND increment Ptc_j by 1

AND add each fp_i to Pfp_{ij}

CASE 2: $fc_j < neg$

THEN Increment Ntc_j by 1

AND add each fp_i to Nfp_{ij}

CASE 3: $Ptc_j \geq min_{ij}$ AND $Pfp_{ij} > pos \times Pfc_j$

THEN Use AND-connective to join premises i and consequent j *to form new rule* nr_{ij}

CASE 4: $nr_{ij} = er_{ij}$

THEN remove nr_{ij} from the list of potential rules

CASE 5: *cor* > threshold of negative evidence

THEN remove nr_{ij} from list of potential rules

CASE 6: *cov* < quota of existing rules

AND *rob* > very high threshold

THEN remove nr_{ij} from list of potential rules

CASE 7: *eff* ≤ 1

THEN Reset *pos* very low and *neg* very high and repeat the algorithm for just consequent j

CASE 8: $sim = 1$

THEN switch rule nr_{ij} for rule er_{ij} in the fuzzy rulebase

CASE 9: $sim = 0$

THEN remove nr_{ij} from the list of potential rules

CASE 10: $nat = 0$

THEN remove nr_{ij} from the list of potential rules

Decrement *pos* and increment *neg* by a small amount δ

Add all nr_{ij} to the fuzzy rulebase

6.4.2 Adapting the algorithm to include Boolean premises.

This algorithm can be adapted to also include Boolean premises. These are variables that take values 1 or 0 corresponding to TRUE and FALSE. In order to adapt the algorithm, two extra columns will be added to the PIT matrix and the NIT matrix for each Boolean premise. Each time a consequent occurs in the example set the value +1 is added cumulatively to the PIT matrix that corresponds to the value of the Boolean variable. A Boolean premise is added to the premises of a potential rule only when the cumulative total for positive occurrences of the Boolean premise is exactly equal to the number of occurrences of the consequent. The cumulative value of the negative occurrences must also be equal to zero. If Pbp_{ij} is the cumulative value for a Boolean premise in the PIT matrix and Nbp_{ij} is the cumulative value for the same variable in the NIT matrix, then this test can be stated more formally as follows:

CASE: $Pbp_{ij} = Ptc_j$ AND $Nbp_{ij} = 0$

THEN Use AND-connective to join Boolean premise i to new rule nr_{ij}

6.5 Testing the behaviour of this technique.

This algorithm was tested using a fuzzy system for X-ray rocking curve analysis (see Chapter 7 for a detailed presentation of this sytem). A set of 59 examples was used to create new fuzzy rules by inductive learning. These examples were taken from good decisions involving single-layer experimental rocking curves of substrate GaAs and layer AlGaAs. This example set is shown in Appendix B. 7 fuzzy variables were used as premise values, each taking 5 linguistic values. 16 Boolean variables were also used as premises, each taking 2 linguistic values. 18 fuzzy consequents were used, each taking 5 linguistic values. The dimensions of the induction test matrices were therefore $(7\times5) + (16\times2) + 2 = 69$ columns and $(18\times5) = 90$ rows. The positive threshold was initialised to 0.95 and the negative threshold was set to 0.125.

11 new rules were created by the basic inductive learning algorithm using no fitness functions. These rules are illustrated in Figure 6.3. This figure also illustrates the point at which rules were removed by the algorithm from the list of new rules. The fitness function for correctness removed 2 rules leaving a total of 8 new rules. For example, in Rule 8 the premises `substrate peak broadening = SOME`, `substrate asymmetry in peak = SOME`, `layer wedge shaped peak = NONE` and `interference fringes = SOME` occurred in the example set when the fuzzy consequent `layer is thick = VERY` also occurred. However, when this rule was processed by the *cor* fitness function it was found that the same premises occurred in

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *interference_fringes* = NONE
THEN *crystal_quality* = FAIRLY good.

RULE 1: Removed by *sim* function.

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *interference_fringes* = NONE
THEN *characteristic_curve* = EXTREME

RULE 2: Removed by *sim* function

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *layer_wedge_shaped_peak* = NONE
AND *interference_fringes* = NONE
AND *intensity_of_layer_peak* = EXTREME
THEN *misorientation_of_substrate* = NONE

RULE 3: Removed by *sim* function

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = SOME
AND *substrate_asymmetry_in_peak* = SOME
AND *layer_wedge_shaped_peak* = SOME
AND *interference_fringes* = SOME
AND *intensity_of_layer_peak* = EXTREME
THEN *misorientation_of_substrate* = SOME

RULE 4: Removed by *sim* function

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *layer_wedge_shaped_peak* = NONE
AND *interference_fringes* = NONE
AND *intensity_of_layer_peak* = EXTREME
THEN *bending_of_substrate* = NONE

RULE 5: Removed by *sim* function

IF *type_of_structure_is_single_layer* = TRUE
AND *number_of_peaks_is_two* = TRUE
AND *relaxed_mismatch_is_high* = FALSE
AND *substrate_peak_broadening* = SOME
AND *substrate_asymmetry_in_peak* = SOME
AND *layer_wedge_shaped_peak* = SOME
AND *interference_fringes* = SOME
AND *intensity_of_layer_peak* = EXTREME
THEN *bending_of_substrate* = SOME

RULE 6: Removed by *sim* function

IF *visibility_of_interference_fringes* = EXTREME
THEN *layer_is_thick* = FAIRLY high

RULE 7: Removed by *cov* function, but accepted by *rob* function. This rule is accepted by the algorithm

IF substrate_peak_broadening = SOME
AND substrate_asymmetry_in_peak = SOME
AND layer_wedge_shaped_peak = NONE
AND interference_fringes = SOME
THEN layer_is_thick = VERY

RULE 8: Removed by *cor* function

IF *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *interference_fringes* = NONE
THEN *layer_is_thin* = FAIRLY high

RULE 9: Accepted by the algorithm

IF *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *layer_wedge_shaped_peak* = NONE
AND *interference_fringes* = NONE
THEN *evidence_of_mismatch* = SOME

RULE 10: Removed by *cor* function

IF *substrate_peak_broadening* = NONE
AND *substrate_asymmetry_in_peak* = NONE
AND *interference_fringes* = NONE
THEN evidence_of_mismatch = FAIRLY high

RULE 11: Accepted by the algorithm

Figure 6.3. 11 rules created by the inductive algorithm, 8 of which are removed by fitness functions: 3 by *cor* (correctness) and 5 by *sim* (simplicity).

a significant number of cases when the fuzzy consequent NOT(layer is thick = VERY) occurred. For this reason, Rule 8 was removed by the *cor* fitness function. Rule 10 was removed for the same reason.

The quota for coverage was set at 5 rules and Rule 7 failed this fitness test. This is because 7 existing rules contained the consequent layer is thick = FAIRLY high. However, this rule was accepted because the value of the *rob* function was greater than 0.975.

A low coverage threshold of zero was set for effectiveness and as each consequent was covered by at least one rule in the existing ruleset, the test for effectiveness was passed without any changes to the positive and negative thresholds.

In 6 of the remaining 9 rules, existing rules covered the same consequents with a subset of these premises. Therefore, these five rules were removed by the *sim* function from the set of new rules. In the case of Rule 1, an existing rule:

```
IF type_of_structure_is_single_layer = TRUE
 AND number_of_peaks_is_two = TRUE
THEN crystal_quality = FAIRLY good
```

covers the same consequent using a subset of the premises. This is Rule [1] in Partition 2 of the fuzzy rulebase (see Appendix C). In the case of Rule 2, the existing Rule [63] covers the same consequent using a subset of the premises. Rule [115] is simpler than Rule 3, Rule [114] is simpler than Rule 4, Rule [6] is simpler than Rule 5 and Rule [5] is simpler than Rule [6]. For these reasons, Rules 1, 2, 3, 4, 5 and 6.

The three rules that remained were presented to the expert Professor D. K. Bowen, who accepted that these rules were sensible. They were therefore added to the existing fuzzy ruleset.

X-ray rocking curve analysis is a changing application domain. A fuzzy system for this application must adapt its rules to deal with new situations and facts. It must also create new rules to deal with new relationships between inputs and outputs to the fuzzy system. This is because the existing rules in the fuzzy rulebase cannot describe all the possible relationships that may be presented to the fuzzy system. The algorithm presented in this chapter demonstrates a method for the automatic creation of sensible fuzzy rules for this application.

6.6 Summary of the behaviour of this technique.

This algorithm creates new relationships between inputs and outputs by inductive learning from examples. These new relationships are presented in the form of new fuzzy rules. Each new rule is tested rigorously using a set of fitness functions. These functions are based on six well-established evaluation criteria for knowledge-based systems. Rules that fail these criteria are removed and only rules that satisfy the evaluation criteria are added to the fuzzy rulebase. This algorithm, combined with the techniques presented in Chapters 4 and 5, will adapt a fuzzy system to deal effectively with a changing application domain. To illustrate this, these techniques and algorithms have all been implemented in a fuzzy system for X-ray rocking curve analysis. This fuzzy system is presented in detail in Chapter 7.

CHAPTER 7. A fuzzy system for X-ray rocking curve analysis.

7.1 Introduction.

In this chapter, a fuzzy system for X-ray rocking curve analysis is presented (Partridge and Tjahjadi 1991, 1994a, 1994b). A fuzzy system can be built in five steps: articulating the problem; developing a strategy for the solution of this problem; deciding which tools to use; implementing the strategy; and testing the solution. However, these steps are not always distinct and there is a large amount of overlap between them. This is typical of AI problem solving, where an initial top-down approach to the problem gives way very easily to amendments and backtracking (D. Partridge 1992). By the time the final solution is implemented, there is a final set of structures and a set of arguments to justify these structures, but these arguments will only have become apparent during the process of building the fuzzy system. For example, the simplified fuzzy rules (presented in Section 5.2) were suitable to the earliest formulation of the problem of X-ray rocking curve analysis. Only when the full complexity of the problem was understood did it become important to use a more sophisticated formulation of fuzzy rules. This formulation uses triangular membership functions (see Section 5.3). For this reason, an alternative format that overlaps many of the five tasks mentioned above is used to present the fuzzy system. In this alternative format, five broad headings are used: the design of the fuzzy system; the description of the fuzzy system; the performance measure for X-ray rocking curve analysis; the implementation of the fuzzy system; and the testing of the fuzzy system.

7.2 The design of the fuzzy system.

The design of the fuzzy system presents an overall strategy for solving the problem of X-ray rocking curve analysis. This section assumes the review of the

problem presented in Chapter 3 and identifies a number of stages in the solution of this problem. A series of techniques or tools are presented to solve each stage in the problem. These include: frames, for describing the experimental and theoretical rocking curves; logic-based variables, which are used to translate these descriptions into a format more suitable for performing deductions; and self-adaptive and self-evaluative techniques, which are used to alter the deductive mechanisms of the fuzzy system to deal with a changing application domain. Arguments are given as to why each of these tools are the most appropriate for the current problem.

7.2.1 Description of the experimental rocking curve.

An adequate description of the experimental rocking curve must be stored. This must include all the important features that will be used to deduce the underlying structure that produced the rocking curve (see Chapter 3). A frame-based representation is the most expressive knowledge representation technique (Partridge 1989). A frame is a data-structure for representing stereotyped situations (Minsky 1981). Frames consist of slots and each slot has a name and a default value attached to it. New values will supersede these default values. Collections of related frames are linked together into what are called frame systems. In a frame system slots and default values are inherited between one frame and another. Frames are extremely pragmatic because knowledge is characterised on the basis of its function or use (Partridge 1989).

Rocking curves have a stereotyped format. They incorporate substrate, layer and satellite peaks and the relationships between each of these (Partridge and Tjahjadi 1991). For this reason, a frame-based representation is the most suitable knowledge

representation technique for the description of rocking curves.

There are two methods that can be used to fill the frame system: direct input of the rocking curve to the frame system; and question and answer sessions with the user. The first method is the most efficient because it automates the data capture. Suitable procedures can be used to prioritise the important features and slots can be filled from direct calculations taken from the curve. The second method uses question and answer sessions with the user to capture the important data from the curve.

7.2.2 Deduction of structural parameters from the experimental rocking curve.

A set of control mechanisms is needed to deduce the structural parameters from the stored description of the experimental rocking curve. In these deductions there are a number of parameters which are imprecise and uncertain. These include such features as asymmetry in the peak and grading of the layer. However, there are also a number of exact parameters, e.g. the number of main peaks. Imprecise and uncertain parameters can be modelled using a set of fuzzy variables. Precise parameters can be modelled using a set of Boolean variables. A fuzzy rulebase that uses both fuzzy variables and Boolean variables in the premises of the rules and a single fuzzy variable in the consequence of a rule can be used to model the deduction of structural parameters from a stored description of an experimental rocking curve. Fuzzy variables are used in the consequences of these rules because the relationship between the description of an experimental rocking curve and the underlying structure is uncertain and imprecise in terms of the deductions that can be made (see Section 3.3).

7.2.3 Deciding which rules to use in a decision.

It is impossible to prescribe for every structure and experimental rocking curve in a fixed set of rules (see Section 3.5). X-ray rocking curve analysis is a changing application domain and self-adaptive and self-evaluative mechanisms must be used in order to change the reasoning processes of the fuzzy system to suit particular types of structures. Fortunately, when experts perform this type of analysis they usually concentrate on one type of structure for a long period of time and then move to a different structure, which they will again concentrate on for a long period (from interview with Professor D. K. Bowen).

Fuzzy rules can be derived from human experts and stored in a fuzzy rulebase. The rules derived in this way can often be contradictory. Some experts can also be experienced in just one type of structure and when the fuzzy system focuses on a different type of structure, the ruleset of that expert will be inadequate. Connection matrices and credibility weights are the most suitable control mechanism for this kind of situation (see Section 4.2). These structures are designed to deal with contradictory knowledge stored in a fuzzy rulebase. They are also designed to alter the focus of attention between different subsets of the rulebase. This is important in a changing application domain.

7.2.4 Adapting the fuzzy system to a changing application domain.

The range of structures that may be presented to the fuzzy system is computationally infinite (see Section 3.5). Therefore, the fuzzy system must shift the focus of attention between different sets of rules. By incrementally altering the credibility weights used in the connection matrices, the focus of attention can be

altered between different sets of rules (see Section 4.3). The meaning of the fuzzy rules can be incrementally altered so that the fuzzy system becomes gradually more effective in dealing with new types of structures (see Chapter 5). Fuzzy rules that are extremely unsuccessful can be hidden using the Unassert function and returned to use by the Assert function (see Section 4.4). In situations where new combinations of input and output variables occur, then new rules will need to be created. The most suitable method for creating new fuzzy rules in a changing application domain is inductive learning from examples (see Chapter 6). To implement this method, each rule element must be encoded as a set of characters or numbers. This encryption also stores the rulebase more efficiently.

7.2.5 Presentation of deductions to the user.

Once the structural parameters are inferred, then these deductions must be used to simulate a theoretical rocking curve. One method is to transfer the structural parameters directly into a simulation program and calculate the theoretical curve immediately from this. However, this method assumes that there are dynamic data links between the fuzzy system and the simulation package. When these data links do not exist, then the output from the fuzzy system must be presented to the user in such a way as to enable the user to enter the structural parameters into the simulation package.

7.2.6 Description of the theoretical rocking curve.

An adequate description of the theoretical rocking curve must be captured and stored by the fuzzy system. This description must be compatible with the stored

description of the experimental rocking curve so that both curves can be compared. For this reason, the frame-based representation is the most suitable to store the theoretical curve.

7.2.7 Comparison of the experimental and theoretical rocking curves.

Comparison of the curves must be based on a set of the most important features. These are the features which are most sensitive to changes in the important structural parameters. These features can be used to calculate a measure of the closeness of matching between the curves.

By measuring the closeness of matching between the two curves, the performance of the fuzzy system in deducing an initial trial structure for simulation is also measured. This measure can be used to update the self-adaptive and self-evaluative control mechanisms and hence to optimise the performance of the fuzzy system over time. The performance measure for X-ray rocking curve analysis is presented in detail in Section 7.4.

7.3 The description of the fuzzy system.

The fuzzy system for X-ray rocking curve analysis uses a set of control structures based on the initial design (see Section 7.2). The control structures and the flow of information between these structures are shown in Figure 7.1. In this figure, the frame system is used to describe the rocking curves and the structural parameters. The logic-based variables are used to transform the knowledge in the frame system into a format more suitable for inferencing using fuzzy rules. The fuzzy rulebase deduces structural parameters from the experimental rocking curve. Connection

matrices and credibility weights decide which fuzzy rules to use in each decision made by the fuzzy system. A record of values for previous decisions is stored and used to update credibility weights and the membership functions of fuzzy premises. This record is also used to inductively learn new fuzzy rules and to Assert/Unassert existing rules. The record of values is derived from a comparison of the experimental and theoretical rocking curves. Each of these control structures is now presented in detail.

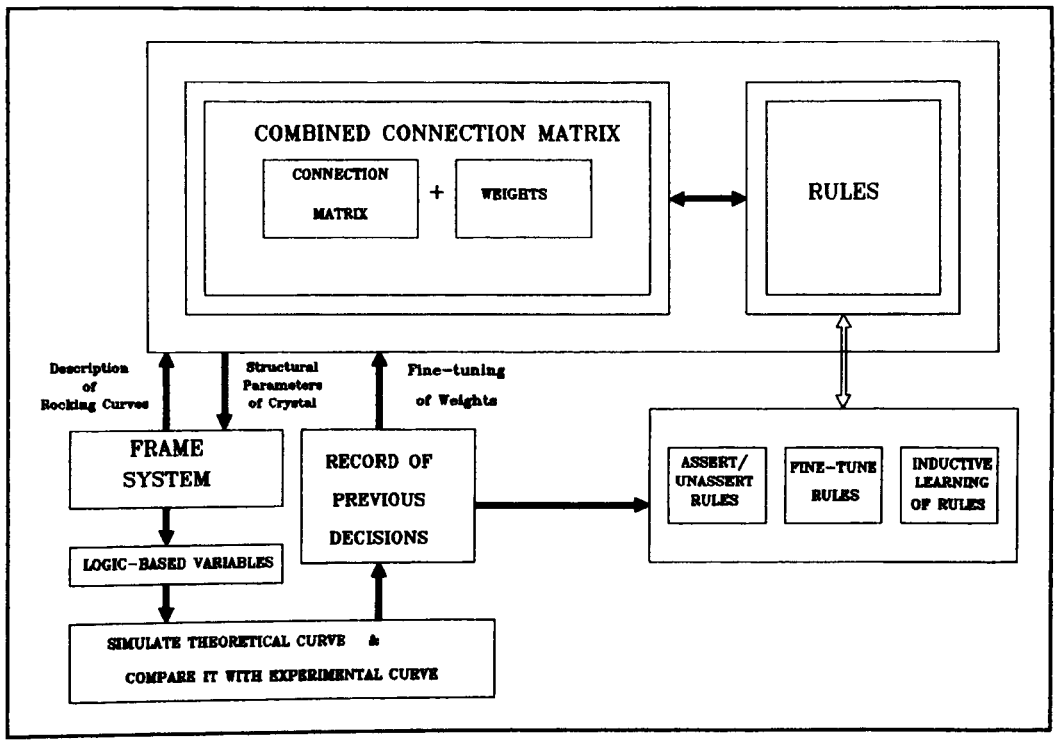

Figure 7.1 The control structures and flow of information in the fuzzy system for X-ray rocking curve analysis.

7.3.1 Frame system.

A frame system is used to describe the experimental rocking curve. This frame system is shown in Figure 7.2.

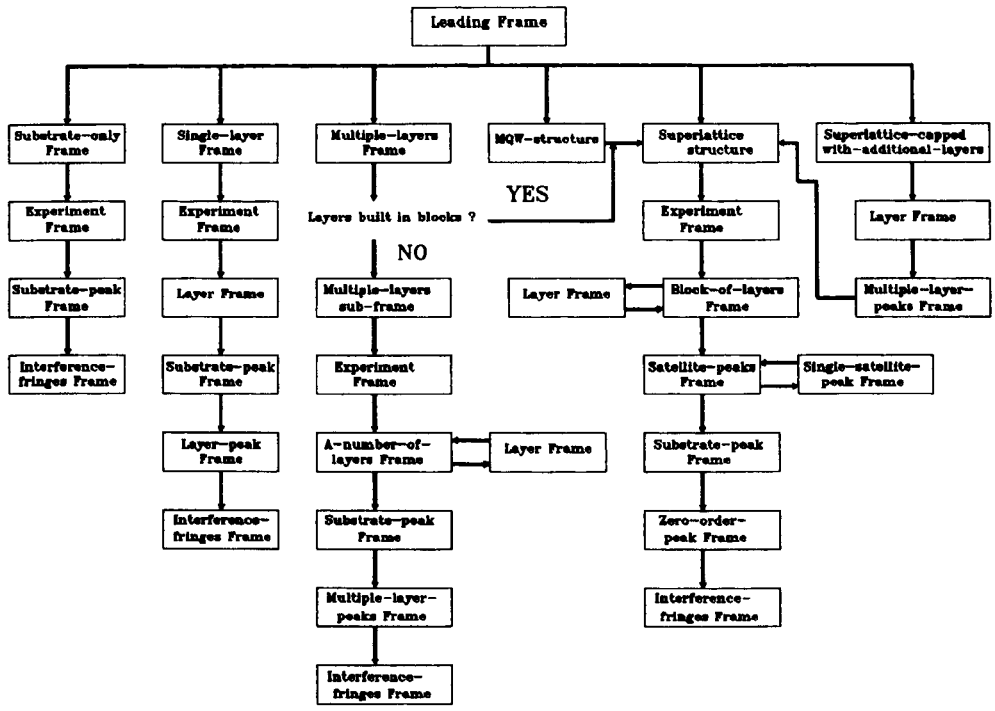

Figure 7.2 Frame system used to describe experimental rocking curves.

The Leading Frame questions the user about which basic type of structure is currently being investigated (see Section 3.6). The fuzzy system then fills a subset of the frame system that corresponds to this type of structure. For example, a superlattice structure will fill the following frames: Leading Frame; Superlattice structure Frame; Experiment Frame; Block-of-layers Frame; a number of Layer Frames, the number appropriate to the number of layers in the block (see Sections 3.6.6 and 3.6.7); Satellite-peaks Frame; two Single-satellite-peak Frames; Substrate-peak Frame; Zero-order-peak Frame; and Interference-fringes Frame. An example of a Substrate-peak Frame is shown in Figure 7.3.

This frame system is built in a tree structure, but the flow of control can move between the branches when this is appropriate. For example, an MQW structure is a less general type of superlattice structure. When an MQW experimental curve is being

described, the particular frame that distinguishes it from a general superlattice structure is filled. Then, the superlattice frames are filled and a number of values typical of MQWs is inherited along this branch of the frame system.

At present, the fuzzy system for X-ray rocking curve analysis is written on a SUN4 under UNIX, while the RADS simulation program (RADS 1992) is written on a PC under MSDOS. Dynamic data links between these two systems do not at present exist. For this reason, it has been decided to use question and answer sessions with the user to fill the frame system.

7.3.2 Logic-based variables.

The knowledge contained in the frame system is transformed into a logic-based format. This is a straightforward process, as frames can be naturally formulated in terms of Zadeh's fuzzy logic notation (Zadeh 1987). For example, the logic-based formulation of the frame in Figure 7.3 is shown in Figure 7.4.

```
SUBSTRATE_PEAK_FRAME
Peak_broadening =>                  Peak_Full_Width_Half_Maximum =>
Integrated_intensity_of_peak => Asymmetry_in_peak =>
(=> indicates a slot, which may be a subframe)
```

Figure 7.3 Example of a Substrate-peak Frame.

```
EDF =   Peak_broadening [Proportion; μ] +
         Peak_Full_Width_Half_Maximum [Number; Width] +
          Integrated_intensity_of_peak [Number; Intensity] +
           Asymmetry_in_peak [Proportion; μ]
```

Figure 7.4 Logic-based format of Substrate-peak Frame.

This formulation is used to describe the 'explanatory database frame' (EDF) for a substrate peak. The frame incorporates a fuzzy measure μ, which is the degree to which a fuzzy variable is compatible with its intended meaning (see Section 2.2.1). For example, asymmetry_in_the_peak is a fuzzy measure μ in the interval [0,1] and the value of μ will describe the level of asymmetry in the peak. It is a fuzzy interpretation of the initial value assigned to the `asymmetry_in_the_peak` slot.

The Boolean and fuzzy logic-based variables can be given values derived from the values stored in the EDFs. The Boolean variables take values in {0,1} while the fuzzy variables take discrete values in [0,1]. These logic-based variables are the inputs to the fuzzy rulebase.

7.3.3 Fuzzy rules.

A partitioned set of fuzzy rules is stored in the rulebase. There are four partitions: the first partition contains 19 rules for substrate only structures; the second partition contains 117 rules for single layer structures and graded single layer structures; the third partition contains 31 rules for multiple layers and graded multiple layers structures; and the final partition contains 76 rules for MQW structures, superlattice structures and superlattices with additional layers top and bottom. The rulebase is partitioned for two reasons: firstly because the fuzzy system runs faster and more efficiently with partitions; and secondly because the rules in the fuzzy rulebase can be naturally split into four heterogeneous partitions.

The fuzzy rules have Boolean variables and fuzzy variables in the antecedent of the rule. The premises are connected using the AND-connective. The consequence of the rule is a single fuzzy variable that describes a structural parameter. In X-ray

rocking curve analysis, inferences made about an experimental rocking curve are used to gather evidence for structural parameters. The structural parameters are imprecise and uncertain, and are therefore best modelled using fuzzy variables. An example of a fuzzy rule for a single layer structure is shown in Figure 7.5. This rule has two Boolean variables and two fuzzy variables in the antecedent. The Boolean variables `type_of_structure_is_single_layer` and `peak_separation_is_low` both take value 1. The fuzzy variables `layer_asymmetry_in_peak` and `layer_wedge_shaped_peak` are defined using triangular membership functions. Both variables have the same term set: {EXTREME, VERY, FAIRLY, SOME, NONE}, but the meanings of these linguistic values are defined using different triangular membership functions. The rule is fired using the method presented in Chapter 2. The fuzzy variable `grading_of_the_layer` uses the same term set but uses different triangular functions. The complete ruleset is shown in Appendix C.

IF type_of_structure_is_single_layer = TRUE
AND peak_separation_is_low = TRUE
AND layer_asymmetry_in_peak = EXTREME
AND layer_wedge_shaped_peak = VERY
THEN grading_of_the_layer = EXTREME

Figure 7.5 Example of a fuzzy rule for a single layer structure.

7.3.4 Connection matrices and credibility weights.

The fuzzy system uses connection matrices and credibility weights to decide which rules to use in a decision. These mechanisms are presented in detail in Chapter

4. Connection matrices allow the heuristics of several experts to be combined together. The rules of four experts are used in the fuzzy system. These experts are: Professor B. K. Tanner of Durham University; Professor D. K. Bowen of Warwick University; Dr. N. Loxley of Bede Scientific Instruments Ltd.; and Dr. C. R. Thomas of Warwick University.

Each expert is given a credibility weight based on experience. The weighting is in the range [0,1]. Credibility weights are incrementally altered over time. A record of values for previous decisions is used to update the credibility weights. Successful decisions will increase the weighting of an expert, unsuccessful decisions will reduce the weighting. In calculating the size of the increments/decrements to the credibility weights the following values were used: the confidence limit for the effectiveness of an expert's ruleset was set at $suc\% = 75\%$; the confidence limit for ineffectiveness was set at $uns\% = 20\%$; the sample size $N=20$; and, in calculating q in Equation (4.1), $\rho = 1$ and $\epsilon = \rho/\kappa$; $\kappa = 20$ (see Section 4.3.5).

7.3.5 Record of previous decisions.

The record of previous decisions contains four sets of histories. In the first set, the recent decisions that involved rules used by a particular expert are stored. That is, whenever one or more rules contained in an expert's ruleset are used in a decision, then a value is stored for that expert. These values are the actual values calculated using the performance measure for X-ray rocking curve analysis (see Section 7.4). Each history records the most recent 20 decisions that used a particular expert's rules. A threshold is set for successful decisions and, in the case of X-ray rocking curve analysis, the threshold is 0.8. This set of histories is called the

history_of_decisions. The history_of_decisions and the threshold for a successful decision are used to calculate the sizes of increments/decrements made to credibility weights (see Section 4.3).

In the second set of histories, a record is kept of recent positive and negative decisions for each fuzzy rule. Each time a decision using a rule is above the threshold for success (i.e. 0.8), the value 1 is recorded. Otherwise, the value 0 is recorded. This set of histories is called the rule_history and is used to invoke the Assert or Unassert functions (see Section 4.4).

In the third set, three histories are recorded for each fuzzy variable. The first history records the actual values that fired a rule when there was a positive decision. That is, a decision that was above the threshold for success. A sample of the 20 most recent decisions is recorded. A second history records those values that fired a rule when there was a negative decision. These are decisions that were below the threshold for success. These pairs of histories are called the variable_histories. The third history is a record of recent increments or decrements made to the triangular membership functions of a fuzzy variable. This history is called the variable_changes. The three histories are used to fine-tune and change the meaning of fuzzy rules (see Section 5.3).

Each time the fuzzy system makes a good decision, the values of the Boolean premise variables, the fuzzy premise variables and the fuzzy consequence variables are recorded. A good decision is one where the outcome of the decision is very high (> 0.95). Values are added to an example set that will be used to create new fuzzy rules using inductive learning from examples (see Chapter 6). A tally of the occurrence of a fuzzy consequence variable in the example set is also recorded.

7.3.6 Simulation of theoretical curve.

In the application area of X-ray rocking curve analysis, the fuzzy system uses its rules to deduce structural parameters. These predict the structure of the semiconductor crystal. The structural parameters are used to derive a theoretical rocking curve which is compared with the experimental curve. To facilitate this, the fuzzy system presents the structural parameters to the user in the form of input screens for the RADS simulation program (RADS 1992). Question and answer sessions with the user are used to input the resulting theoretical rocking curve. This curve is stored in a frame system and then the important descriptive variables are transformed into a logic-based format. A subset of the frame system for the experimental rocking curve can be used to describe this theoretical curve. This subset will include those parameters that describe the shape of the theoretical rocking curve. The experimental conditions will be the same as for the experimental rocking curve. The filled frames can be translated into a logic-based format for comparison of the two curves.

7.3.7 Comparison of experimental and theoretical rocking curves.

The experimental and theoretical rocking curves are compared together. The comparison produces a performance measure in the range [0, 1]. A low value means there is little matching between the curves, a high value close to 1 means that the rocking curves are very similar in all their important features. This performance measure is presented in detail in the next section.

7.4 The performance measure for X-ray rocking curve analysis.

The performance measure for X-ray rocking curve analysis is called the Outcome of the Decision *O*. The performance measure was derived after discussion with an expert in X-ray rocking curve analysis (from interview with Professor D. K. Bowen). This performance measure is used by the self-adaptive and self-evaluative mechanisms of the fuzzy system. These mechanisms include updating a set of credibility weights used in connection matrices (see Chapter 4); masking or Unasserting fuzzy rules (see Chapter 4); fine-tuning and altering the membership functions used in the antecedents of fuzzy rules (see Chapter 5); and creating new fuzzy rules using inductive learning from examples (see Chapter 6). The formulation of each of these mechanisms depends on an adequate performance measure.

The Outcome of the Decision *O* is a measure of the degree of matching between the experimental rocking curve and the theoretical curve derived from a set of structural parameters deduced by the fuzzy system (Partridge and Tjahjadi 1994a). The basis for the comparison between the two curves is a two-stage process involving qualitative and quantitative analyses.

7.4.1 Qualitative analysis.

The features of the curves are prioritised on the basis of importance. The position, height, width and shape of the major peaks are given the highest priority. A discrepancy between the experimental and theoretical curves on the basis of these criteria would involve serious re-evaluation of the fuzzy system's ruleset. The second most important set of criteria are the minor peaks. Curves are checked to see if the minor peaks are present and whether these peaks are correct for the same factors as

the major peaks. The third most important set of criteria are the background levels between peaks. Finally, the least important set of features are the tails of the curves.

Initially, evaluation is concentrated on the first set of criteria. When the fuzzy system functions satisfactorily on this basis, the focus of attention is moved to the other sets of criteria.

7.4.2 Quantitative analysis.

The quantitative analysis measures the degree to which the fuzzy system is functioning satisfactorily. From the qualitative analysis, two ordered sets of features are obtained, one for the experimental curve *(E)* and one for the theoretical curve *(T)*. In Equation (7.1), O gives a measure for the outcome of the current decision:

$$O = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\min(E_i, T_i)}{\max(E_i, T_i)} \right) \quad \text{when NOT}(E_i = 0 \text{ and } T_i = 0)$$
$$= \quad 1 \quad \text{otherwise} \qquad (7.1)$$

where n is the number of features being tested. O is always in the interval [0, 1]. The higher the value of O, the closer the match between the experimental and theoretical curves. A threshold is set for successful decisions and, whenever the value of O is above or equal to this threshold, the level of matching between curves is judged to be significantly high. This is called a successful decision. The threshold has been set at the value 0.8.

7.5 The implementation of the fuzzy system.

The fuzzy system for X-ray rocking curve analysis is implemented on a SUN 4, using POP11 Flavours (POPLOG 1991). The POP11 language is used because its

object-oriented facility, called Flavours, allows the development and testing of different strategies in a short period of time. Generic objects are defined. The most important of these are slots, frames, fuzzy variables, rules, connection matrices, the combined connection matrix and a control object called the fuzzy system. These objects encapsulate the important variables and procedures associated with each of these concepts. By sending messages to an instance of the object, methods are invoked which implement the functions outlined in the strategy for the fuzzy system (see Sections 7.2 - 7.4, above). The inheritance facility of this object-oriented language means that the generic objects can be specialised and fine-tuned to suit each new specification of the problem. This flexibility is the main reason why POP11 was used in the development of this fuzzy system.

POP11 is part of the POPLOG environment and is available for the SUN under UNIX. The RADS simulation program is written for use on a PC under DOS. Although there are object-oriented languages available under DOS, it was decided to develop the fuzzy system on a SUN4 under UNIX. This is because, the self-adaptive mechanisms used in the fuzzy system (see Chapters 4 - 6) need a large amount of memory as well as a fast machine in order to process the records of decisions, construct connection matrices, invoke the ruleset for numeric representations of rule components, calculate the outcome of decisions, generate new rules by inductive learning and test each new potential rule. It was especially important in the development stages of the fuzzy system to concentrate on testing the various control mechanisms without undue concern for memory space or the speed of processing. The SUN4 is a suitably fast machine that has a large RAM.

7.5.1 The fuzzy system program.

The fuzzy system for X-ray rocking curve analysis is invoked using either of four commands. These commands correspond to four separate functions: initialisation of the fuzzy system; running the fuzzy system; generation of new fuzzy rules by inductive learning; and adding new rules to the fuzzy system. The complete code for the fuzzy system is stored in 12 files. These files are listed in Table 7.1. The code contained in these files can be found in Appendix D.

File name	Description of program file
initial.p	This program initialises the main variables in the fuzzy system.
fuzzy.p	The main program that runs the fuzzy system.
setfuzz.p	This program sets up the fuzzy consequent variables.
rules1.p	This program sets up the fuzzy rules in the first partition of the rulebase.
rules2.p	This program sets up the fuzzy rules for the second partition of the rulebase.
rules3.p	This program sets up the fuzzy rules for the third partition of the rulebase.
rules4.p	This program sets up the fuzzy rules for the fourth partition of the rulebase.
rules.p	This program sets up the rulebase from the numerical representation.
fuzzout.p	This program is used to calculate defuzzified values for structural parameters
thick.p	This program uses the integrated intensities to calculate an estimate of layer thickness.
induce.p	This program performs inductive learning of new fuzzy rules from examples.
editrule.p	This program is a simple rule editor for adding new rules to the fuzzy rulebase.

Table 7.1 List of files used to store the complete code for the fuzzy system program.

7.5.2 The initialisation of the fuzzy system.

In the initialisation of the fuzzy system, all the important global variables are

declared and given initial values. This program is run once only, that is when the fuzzy system is first started. For this reason, this program is kept separate from the main fuzzy system program. To invoke this function the user types: **pop11 +initial**, at the UNIX command line. This command loads the POPLOG environment, loads the object-oriented facility, loads the program file **initial.p**, creates an instance of the object FUZZY_SYSTEM and sends the message *initialize_the_system_variables* to this instance.

The file **initial.p** contains two objects: the CONNECTION_MATRIX object and the FUZZY_SYSTEM object. The message *initialize_the_system_variables* invokes a method that creates a numerical representation of the ruleset, sets up the membership functions, initialises the histories and creates the connection matrices for each of the experts. To create the connection matrices, the FUZZY_SYSTEM object creates an instance of the CONNECTION_MATRIX object. A number of other variables are also declared and initialised. These include control variables for Equation (4.1) and for the inductive learning algorithm (see Chapter 6). Finally, each of these variables is stored to an external data file for use by the main fuzzy system program (**fuzzy.p**). The complete list of external data files is shown in Appendix E.

7.5.3 Running the fuzzy system.

The main fuzzy system program is contained in the program file **fuzzy.p**. This program implements the design and description of the fuzzy system presented in Sections 7.2 - 7.4. However, because the generation of new fuzzy rules by inductive learning occurs only at irregular intervals, this function has been separated from the rest of the program (see Section 7.5.4). To invoke the main program the user types:

pop11 +fuzzy, at the UNIX command line. This command loads the POPLOG environment, loads the object-oriented facility, loads the program file **fuzzy.p**, creates an instance of the object FUZZY_SYSTEM and sends the message *run_the_system* to this instance.

The object FUZZY_SYSTEM stored in the file **fuzzy.p** is a different and a much more complex object than the one stored in **initial.p**. The file **fuzzy.p** contains 31 generic objects, the most important of which are: SLOT; FRAME; FRAME_SYSTEM; STRUCTURAL_PARAMETER_FRAME; FUZZY_VARIABLE; RULE; CONNECTION_MATRIX; COMBINED_MATRIX; and FUZZY_SYSTEM. The other objects in this file are specialisations of these more generic objects. This program file also contains a large number of globally declared variables that act as logic-based variables for the experimental rocking curve, the theoretical rocking curve and the structural parameters. Eight external program files are loaded by the **fuzzy.p** program. These programs are kept external so that they can be easily changed without affecting the main fuzzy system program. This allows a programmer to easily alter the numerical representation of rule components or the sets of premise and consequent variables to the fuzzy system.

The message *run_the_system* invokes a method that makes an instance of the object FRAME_SYSTEM and asks the leading question (see Section 7.3.1). This leading question decides which type of structure is currently being analysed. A subset of the frame system which corresponds to this structure is filled using question and answer sessions. Then the frames are translated into a logic-based format using the set of globally declared variables (see Section 7.3.2). The external program file, **setfuzz.p**, is loaded. This program sets up the fuzzy consequent variables, including

the membership functions and the variables used to calculate the defuzzified output values.

An instance of the object COMBINED_MATRIX is created and the combined connection matrix is loaded from an external data file. However, if the credibility weight of an individual expert has changed since the last decision was made by the fuzzy system, then a new combined matrix is calculated. To do this, instances of CONNECTION_MATRIX are created for each expert and methods invoked to calculate the new combined matrix (see Chapter 4).

Depending on the type of structure being analysed, a partition of the fuzzy rulebase is loaded using the numerical representation of the rule elements. There are four partitions, stored respectively in the program files **rules1.p**, **rules2.p**, **rules3.p** and **rules4.p**. A further program file **rules.p** is used to construct the ruleset from these values. Rules are stored in a $3 \times n$ array, where n is the number of rules in the partition (see Section 7.3.3).

The message *make_deductions* is sent to the object COMBINED_MATRIX. This makes a list of those rules with the highest weights in the combined connection matrix. The objects RULE and FUZZY_VARIABLE are then used to fire the rules layer by layer and calculate the defuzzified output values using the centroid defuzzification method (see Section 2.2). The external program file **fuzzout.p** is used to store these consequences of the fuzzy rules.

An instance of the STRUCTURAL_PARAMETERS_FRAME object is created and this frame is filled. The calculations are made of values such as experimental and relaxed mismatch, layer thickness, the period of a superlattice and period dispersion. Then the structural parameters are displayed in such a way as to allow the user to

input these values to the RADS (RADS 1992) simulation program (see Section 7.3.6).

Question and answer sessions with the user are used to fill an instance of FRAME_SYSTEM for the theoretical rocking curve. This description is then transferred into a logic-based format and the outcome of the decision O is calculated using Equation (7.1). The record of decisions and other histories for the experts, the rules and the fuzzy variables (see Section 7.3.5) are updated and these histories are stored to external data files (see Appendix E). If the value of the outcome of the decision O is very high (greater than 0.95), then the values taken by each fuzzy premise variable, each Boolean premise variable and each fuzzy consequence variable are added to the example set used for inductive learning.

The self-adaptive and self-evaluative mechanisms are checked and if the correct conditions occur then the credibility weight of an expert can be changed (see Section 4.3), a rule can be ASSERTed/UNASSERTed (see Section 4.4) or the membership function of a fuzzy premise variable can be fine-tuned (see Section 5.3). Methods in the FUZZY_SYSTEM object are used to calculate the size of these changes.

Finally, the user is asked if they wish to analyse another crystal and the process is repeated until the answer to this question is NO.

An example session using this program is shown in Appendix F. This session is for a single layer structure of layer AlGaAs on a GaAs substrate.

7.5.4 Generation of new fuzzy rules by inductive learning.

The generation of new fuzzy rules by inductive learning uses an example set taken from good decisions made by the fuzzy system. This learning occurs only at irregular intervals and so this function is separated from the main fuzzy system

program. To invoke this function the user types: **popl1 +induce**, at the UNIX command line. This command loads the POPLOG environment, loads the object-oriented facility, loads the program file **induce.p**, creates an instance of the object FUZZY_SYSTEM and sends the message *create_new_rules_by_induction* to this instance. Two parameters are also sent with this message, namely the minimum sample size before induction is allowed to occur and the number of occurrences of a particular consequence in the example set before a new rule is allowed to be generated.

The file **induce.p** contains two objects: the CONNECTION_MATRIX object and the FUZZY_SYSTEM object. The message *create_new_rules_by_induction* invokes a method which loads the example set from the data files (see Appendix E), builds the PIT and NIT matrices (see Section 6.2), checks for potential rules in the PIT matrix, tests the fitness function for correctness using the NIT matrix (see Section 6.3.1), then tests each potential rule for effectiveness (see Section 6.3.4), simplicity (see Section 6.3.5), coverage (see Section 6.3.2), robustness (Section 6.3.3) and naturalness (see Section 6.3.6). Finally, potential rules that satisfy these fitness tests are added to the rulebase and the control variables are changed accordingly. An example session using this program is shown in Appendix G. This is the session used to test the inductive learning for X-ray rocking curve analysis (see Section 6.5).

7.5.5 Adding new rules to the fuzzy system.

In adding new rules to the fuzzy system, the input variables and output variables are displayed to the user. The user chooses variables and values for these variables. In the case of a fuzzy variable, the values are taken from the term set

{EXTREME, VERY, FAIRLY, SOME, NONE}. In the case of a Boolean variable, the values are taken from the set {TRUE, FALSE}.

This simple user-interface uses those input variables and structural parameters that have already been incorporated into the user-interface of the main fuzzy system program. For the user to define new variables, and construct rules that use these new variables, a new user-interface would need to be created that captures values for these variables. Then, if the user defines new output variables, that is new structural parameters, then the simulation program (RADS 1992) would need to be adapted to simulate for these new variables. In fact, in the case of new fuzzy variables, the user would have to define new membership functions for the new variables, define new questions to capture the appropriate values for these inputs and interpret the new outputs in terms that can be keyed into the screens of the simulation program. Therefore, it seems more appropriate that the fuzzy system for X-ray rocking curve analysis should capture the most appropriate set of variables and fine-tune the system on the basis of these, rather than to radically recreate the fuzzy rulebase by defining completely new sets of variables and rules. For this reason, it was decided to write a very simple rule editor that allowed the user to add new permutations of variables and values using the existing set of input and output variables.

To invoke this function the user types: **pop11 +editrule**, at the UNIX command line. This command loads the POPLOG environment, loads the object-oriented facility, loads the program file **editrule.p**, creates an instance of the object FUZZY_SYSTEM and sends the message *create_new_fuzzy_rules* to this instance. The file **editrule.p** contains two objects: the CONNECTION_MATRIX object and the FUZZY_SYSTEM object. The message *create_new_fuzzy_rules* invokes a method that

allows the user to construct new rules from a series of simple menus, checks whether the rule already exists and adds the rule and changes the important variables accordingly. An example session using this program is shown in Appendix H.

7.6 The testing of the fuzzy system.

The fuzzy system is tested using 100 sample decisions for a single layer structure with layer AlGaAs and substrate GaAs. Each of the self-adaptive mechanisms are tested and the sensitivity of these changes is also monitored. A set of 59 example instances are generated from the 100 sample decisions and this example set is used to generate new fuzzy rules by inductive learning.

Initially, tests were performed on a cross section of typical crystal structures, but it was found that, in jumping randomly from one type of structure to another, the fuzzy system did not converge in a manner suitable for analysing batches of similar structures. Also, the behaviour simulated in this manner was untypical of normal rocking curve analysis and therefore produced irrelevant results. Hence, a batch of single-layer experimental rocking curves, of substrate GaAs and layer AlGaAs, was run through the fuzzy system. The experimental curves were generated using the RADS (RADS 1992) simulation program and compared with the theoretical curves derived from the structural parameters inferred by the fuzzy system. The behaviour of credibility weights, rules, and fuzzy variables was monitored. An example set for inductive learning was also created (see Appendix B). The fuzzy system was tested on the basis of the most important set of curve features (see Section 7.4), that is, that the major peaks were modelled correctly in position, height, width and shape. A threshold of 0.8 was set for a successful decision. This threshold was reached after

discussion with the domain experts. A graph of the performance of the fuzzy system in 100 sample decisions is shown in Figure 7.6. In 76% of cases, the fuzzy system produced correct decisions. In the other 24%, the experimental curves were derived

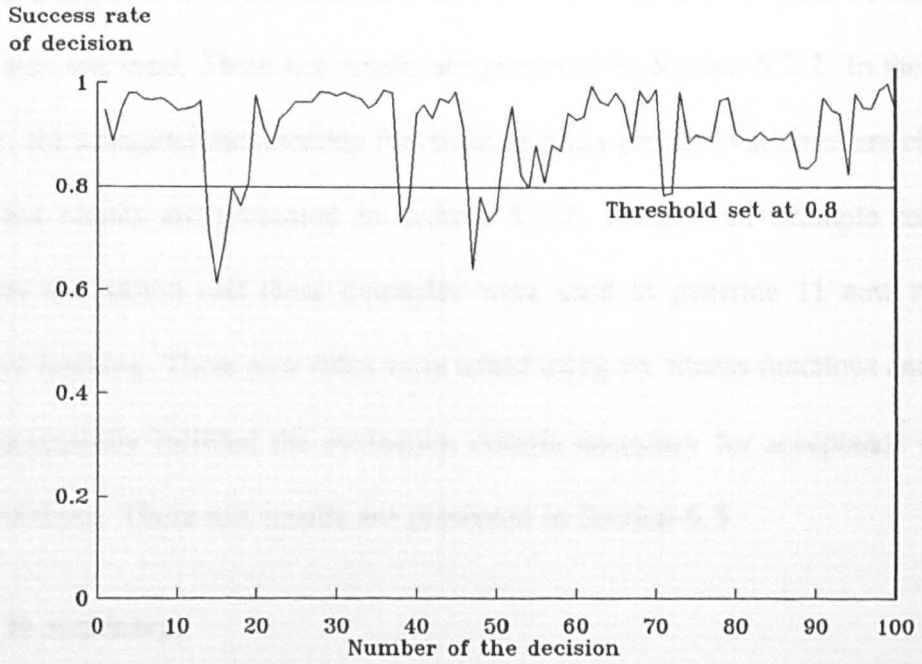

Figure 7.6 Performance of the fuzzy system in 100 sample decisions on single layer experimental rocking curves, of substrate GaAs and layer AlGaAs.

from structures that human experts would find extremely difficult. These were curves with extremely thin or thick layers, with extreme grading or relaxation etc. The majority of such decisions were just below the 0.8 threshold. In fact, in these cases the fuzzy system proposed simulations that would actually be acceptable by experts even though they were below the 0.8 threshold. For example, a layer which is less than 0.5 microns thick will result in a layer peak of very low intensity. In fact, there may be no layer peak whatsoever. Deductions based on this kind of curve will be less valid than those based on more straightforward curves.

Tests showed that the fuzzy system changed its focus of attention by altering

the credibility weights of the experts. These test results are presented in Section 4.3.5. No rules were Unasserted during the 100 sample decisions, implying that the ruleset elicited from the experts is of a high standard. The meaning of the fuzzy rules was changed using two different methods. In the first method a rulebase of simplified fuzzy rules was used. These test results are presented in Section 5.2.2. In the second method, the triangular membership functions of fuzzy premise variables are changed. These test results are presented in Section 5.3.7. Finally, an example set of 59 instances was stored and these examples were used to generate 11 new rules by inductive learning. These new rules were tested using six fitness functions and 3 new rules successfully fulfilled the evaluation criteria necessary for acceptance into the fuzzy rulebase. These test results are presented in Section 6.5.

7.7 In summary.

The fuzzy system presented in this chapter has been shown to infer structural parameters from descriptions of experimental rocking curves. When these structural parameters are used to simulate a theoretical curve the resulting curve has been shown to closely match the original experimental one, except in cases where experts would have difficulty in making inferences. However, by including the self-adaptive and self-evaluative mechanisms presented in Chapters 4 - 6, the fuzzy system has been shown to incrementally improve its performance by focusing on more appropriate rules, by fine-tuning existing rules so that these rules are more accurate given recent inputs and outputs to the system and by generating new fuzzy rules by inductive learning to deal with new situations and facts presented to the fuzzy system. The implemented fuzzy system for X-ray rocking curve analysis is one that can adapt its knowledge to a changing application domain.

CHAPTER 8. Conclusions

8.1 Introduction.

A fuzzy system that models a changing application domain must alter its set of fuzzy rules (see Section 2.3). These rules must be changed so that the rulebase will incrementally follow any changes that occur in the relationships between the inputs and outputs to the fuzzy system. These relationships are modelled by a continuous function (Kosko 1993b). Therefore, the task in adapting a fuzzy system for a changing application domain is reduced to fine-tuning and changing a set of fuzzy rules so that the behaviour of the rulebase follows the changes made to a continuous function.

In this thesis, three methods are proposed for altering a set of fuzzy rules: shifting the focus of attention between different sets of rules; fine-tuning and changing the meaning of individual fuzzy rules; and generating new fuzzy rules from an example set taken from recent good decisions made by the fuzzy system.

Three new techniques are presented for implementing each of these methods: altering the credibility weight of an expert and thereby shifting the focus of attention of a fuzzy system towards those rules that were successful in the recent past (see Chapter 4); fine-tuning and changing the membership functions of fuzzy variables used in the premises of rules and thereby altering the meaning of the fuzzy variables and, hence, the meaning of the rules (see Chapter 5); and, thirdly, generating new fuzzy rules by inductive learning from examples (see Chapter 6). Arguments are presented as to why these new techniques are the most suitable for changing application domains and each of the techniques is tested using a fuzzy system for X-ray rocking curve analysis.

X-ray rocking curve analysis is a changing application domain (see Chapter 3). The continuous function that maps inputs to outputs changes over time and a set of

fuzzy rules that successfully estimates this relationship for one set of structures will be unsuccessful for a different set. Therefore, this application is a suitable one for testing the new techniques presented in Chapters 4 - 6. A fuzzy system has been built for this application (see Chapter 7).

The fuzzy system for X-ray rocking curve analysis uses the following knowledge representation techniques: frames; a logic-based format for these frames; fuzzy rules; a numerical representation of these rules; connection matrices and credibility weights; Assert/Unassert functions; an algorithm for fine-tuning fuzzy rules; and rule generation by inductive learning from examples (see Section 7.3). 100 experimental rocking curves were presented to the fuzzy system (see Section 7.6) and each of the three techniques was tested. These tests showed that the fuzzy rulebase was incrementally altered and adapted to suit the changes in the application domain.

8.2 Benefits of the techniques presented in this thesis.

In this section, five benefits are discussed for introducing three new self-adaptive techniques for fuzzy systems.

8.2.1 The range of applications of fuzzy systems is expanded.

The most important benefit of the techniques presented in this thesis is that the range of application domains for fuzzy systems can be expanded to include applications that change over time. For example, fuzzy systems can be developed for applications such as economic forecasting, customised user-interfaces and epidemiological studies. These applications can change over time and a set of deductions that solves one problem can be useless for the solution of a quite similar

problem in the same domain. By regularly feeding the results of decisions back into a fuzzy system and using the three techniques presented in this thesis, the rules in the fuzzy system can be adapted to suit the most recent behaviour in the application.

8.2.2 Automate the analysis of X-ray rocking curves.

The techniques presented in this thesis have been used to solve a particular problem in the area of X-ray rocking curve analysis. Thus, it has been shown that by adapting the fuzzy rulebase, the performance of the fuzzy system for X-ray rocking curve analysis has been optimised. Optimisation has been measured in two ways: excluding negative decisions from the behaviour of the fuzzy system; and generating new fuzzy rules that will make good decisions given an example set taken from recent good decisions made by the fuzzy system. Therefore, the obvious benefit has been the development of a successful fuzzy system for the application of X-ray rocking curve analysis.

8.2.3 A fuzzy system can be implemented in a changing application domain.

By testing each of the techniques using the fuzzy system for X-ray rocking curve analysis, it has been shown that each technique can be successfully implemented in a real-world application that changes over time. It has also been shown that the program that implements this fuzzy system is, in its final form, efficient in terms of size and speed. The complete code fits on a single 3½ inch (1.4 MBytes) floppy disk and a single session - including the user description of the experimental rocking curve, the deduction of structural parameters, the description of the theoretical curve, the calculation of the outcome of the decision, the recording of histories and the

incremental altering of the control structures - can occur in 5-10 minutes of the user's time.

8.2.4 Combination of inductive learning and a number of techniques more associated with genetic algorithms.

The algorithm for inductive learning from examples combines the intentional statistical-based methods of inductive learning with the more abstract techniques of genetic algorithms. These abstract techniques include mapping rule elements to a numerical representation and using fitness functions to test new rules generated by the inductive learning. This combination of techniques results in the generation of rules that are succinct and relevant, in terms of the rule elements used, and that can be justified explicitly in terms of well-established evaluation criteria for knowledge-based systems.

8.2.5 Introduction of techniques that can be widely applied.

A number of the techniques introduced in this thesis are applicable to intelligent systems other than fuzzy systems. In particular, the technique of incrementally altering the credibility weights of experts (see Chapter 4) can be used to alter the behaviour of production systems that use Boolean variables in the antecedents and consequents of rules. It has also been shown (see Section 6.4.2) that the algorithm for inductive learning from examples (see Chapter 6) can be adapted to generate new rules that use Boolean variables. Thus, one other benefit has been the introduction of new techniques that can also be applied outside the area of fuzzy systems.

8.3 Limitations of the techniques presented in this thesis.

In this section three limitations of the three self-adaptive techniques for fuzzy systems are discussed.

8.3.1 No automatic generation of new input and output variables.

One limitation of these techniques is that they do not allow for the automatic generation of new input and output variables. Thus, when a change occurs in an application domain and that change is created by the introduction of a new parameter that has not previously been legislated for by the fuzzy system, then in the current formulation there is no way to detect the existence of this new parameter. This problem indicates a gap in the underlying knowledge of the fuzzy system. The same gaps can occur in the decision-making of a human expert. This does not mean that the problem goes away, but it means that the solution of the problem is as intractable as it is in the case of human experts.

8.3.2 Generates large number of data files.

The second limitation is that each of these techniques involves the recording of a large number of histories that are stored in data files. This is necessary so that each time the fuzzy system is loaded, the stored histories which affect the self-adaptive mechanisms will include the previous sessions.

8.3.3 The three techniques operate independently.

In the current formulation, each of the three techniques functions separately. That is, evidence is gathered incrementally for changes to the credibility weights, to

the membership functions of fuzzy premises and for the generation of new fuzzy rules. The underlying assumption in this framework is that the behaviour of each technique is independent and will approach equilibrium over time. In tests on 100 experimental rocking curves in a fuzzy system for X-ray rocking curve analysis, it has been shown that equilibrium is reached for each of the three techniques. These separate equilibriums are reached even though the three techniques are working in tandem. However, the effects of one self-adaptive technique on a second technique has not been studied in depth. For example, the membership function of a fuzzy premise of a rule can be incrementally changed while evidence is still being gathered for a change to an expert's credibility weight. The question arises as to whether the credibility weight of the expert should be altered while the rules used by that expert are also changing. If the techniques are not independent then the main effect will be that a false equilibrium, that is above or below the true equilibrium value, is reached. In time, the effects of the new rules or the new credibility weights will incrementally alter the erroneous equilibrium. Therefore, by integrating the tests for each technique and incorporating weightings for dependent effects, the self-adaptive techniques can be made more efficient.

8.4 Future work.

The three new techniques presented in this thesis have been shown to incrementally improve the performance of a fuzzy system in a changing application domain. However, there are a number of areas that could also be investigated in order to improve the self-adaptive techniques or the fuzzy system for X-ray rocking curve analysis.

8.4.1 Statistical analysis of the effects of different thresholds and sample sizes.

The first of these areas is a statistical analysis of the effects of different thresholds and sample sizes on the values calculated for the outcome of the decision *O*. By automating analysis of different values for the confidence limits *suc%* and *uns%*, the sample size *N* for altering credibility weights and the size of the increment κ (in Section 4.3), of the sample size *N* for fine-tuning rules (in Section 5.3) and the positive and negative thresholds for consequent values and the minimum sample size min_{ss} for generating rules (in Section 6.2), values could be derived for each of these parameters that will optimise the outcome of the decision *O* for a sample of decisions.

8.4.2 Altering the values of parameters to optimise the outcome of decisions.

As a result of this analysis, a statistical-based algorithm could be introduced that alters the values of these parameters over time in such a way as to optimise the outcome of the decision *O*. This optimisation would be on the basis of the most recent inputs and outputs to the fuzzy system. The behaviour of this algorithm would be similar to the behaviour of the fine-tuning algorithms (presented in Chapter 5).

8.4.3 New threshold used to invoke the inductive learning algorithm.

A further area that could be investigated is the introduction of a new threshold. This threshold would be a sample mean taken from a sample of recent values for the outcome of the decision *O*. This value could be used to invoke the inductive learning algorithm. The rationale behind this investigation would be that when the value of *O* is always high, then there would be no need to use the inductive learning algorithm to generate new rules. However, once the mean value of *O* in a recent sample of

decisions dropped below a threshold, then it would be important to alter the fuzzy rulebase in such a way as to raise the mean value of O again. The statistical methods, mentioned above, could be used to investigate the level of this new threshold.

8.4.4 Deriving a fitness function for information content.

A further area that could be investigated is the development of a fitness function that measures the information content of new fuzzy rules. Although this evaluation criterion was not mentioned in the criteria of (Guida and Maura 1993), a measure for this criterion was developed (by Gaines and Shaw 1986). The main problem with deriving a fitness function for information content is that such a measure must be in terms of the domain theory of the application. For example, a fitness function that chooses a rule that optimises matching between the broadness and asymmetry of the experimental and theoretical curves in X-ray rocking curve analysis will have a high information content. However, the measure that works in this case will be very unlikely to be satisfactory for applications such as economic forecasting or customised user-interfaces. Therefore, this investigation should be concentrated on deriving a fitness function that is suitable for X-ray rocking curve analysis alone.

8.4.5 Integration of tests for the self-adaptive techniques and weight for dependent effects.

As mentioned above (see Section 8.3.3), a further area that could be investigated is the integration of the tests used for each of the self-adaptive techniques and the incorporation of weights for dependent effects. If a methodology for investigating these effects could be developed, then testing for independence or

dependence of the techniques could be performed methodically for each application domain.

8.4.6 Dealing with exceptions to rules.

One area that could also be investigated is how to deal with exceptions to rules. There is at present no mechanism to deal with a particular set of input/output relationships that is an exception to the more typical behaviour presented to the fuzzy system. The default reasoning (of Reiter 1980) could be used to develop techniques to deal with these types of cases.

8.4.7 Dynamic data links between the fuzzy system and RADS.

One important area that could be investigated is dynamic data links between the fuzzy system for X-ray rocking curve analysis and the RADS simulation program. This would allow the automatic capture of experimental and theoretical rocking curves by the frame system of the fuzzy system and also allow the passing of structural parameters directly from the fuzzy system into RADS. At present the fuzzy system is written in POP11 on a SUN4 under UNIX. RADS is written on a PC under MSDOS. However, given that the development process of the fuzzy system is now complete, a succinct version of the fuzzy system could be written in an object-oriented programming language for a PC under MSDOS. In this process, dynamic data links could be incorporated into the fuzzy system program.

8.4.8 More rigorous testing of the fuzzy system.

The testing of the fuzzy system for X-ray rocking curve analysis could be

extended in two directions. In the first case, a very large number of one type of structure (> 1000) could be presented to the fuzzy system. In this way, the long-term behaviour of the system could be tested in terms of curve features that are of lower priority (see Section 7.4.1) and mechanisms that deal with exceptions to rules (see Section 8.4.6) could also be investigated. Thus, a behaviour pattern that is rare but important could be investigated and accounted for in the fuzzy rulebase. In the second case, testing could be extended to a wider range of structures so that the different partitions of the fuzzy rulebase could be more fully tested.

8.4.9 Extending the behaviour of the fuzzy system for X-ray rocking curve analysis.

Finally, a set of fuzzy rules could be developed for inferring second, third and higher order theoretical rocking curves. These new rules could be used to compare the description of the experimental rocking curve with the description of the first theoretical curve in order to infer new structural parameters. These could then be used to simulate a second theoretical curve. Later, this behaviour could be extended to infer structural parameters for higher order theoretical curves. By this means, the range of application of the fuzzy system could be extended to cover the complete process of X-ray rocking curve analysis.

8.5 In conclusion.

The development of fuzzy systems in changing application domains is an on-going process. In dealing with these kinds of applications it is important not to circumscribe the range of behaviours that are expected of the fuzzy system. For this

reason, although it is important to introduce new self-adaptive techniques that deal with these kinds of domains, it is also important to test these techniques, to identify new problems and to develop new techniques to solve these problems. In all these efforts, it is particularly important to be pragmatic. This means that instead of attempting to develop an ideal fuzzy system that solves all problems perfectly, it is more important to solve all the important problems adequately. Adequacy in this context must be defined explicitly in terms of the application domain.

To this extent, this thesis introduces three new self-adaptive techniques that allow fuzzy systems to make deductions in changing application domains. These techniques have been demonstrated using a fuzzy system for X-ray rocking curve analysis. Although the techniques presented here are generic, the implementation of these techniques for the specific application of X-ray rocking curve analysis means that the solution of this problem is the primary aim of the fuzzy system. The side-effect is the demonstration of the optimisation of performance that is directly attributable to the three self-adaptive techniques. In presenting these techniques applied to a real-world problem, it is therefore important to identify new problems and complications that have occurred in the development of the fuzzy system for this application. These problems are presented in the section, Future work (see Section 8.3).

APPENDICES

APPENDIX A

Test data used to monitor the behaviour of Equation (4.1) (see Section 4.3.5).

100 sample decisions are used. 80 changes occur in the credibility weight of expert Professor D. K. Bowen. The values of p, q, the function f and the actual credibility weight of the expert are monitored over these 80 changes.

Number of change	p value	q value	value of f	credibility weight of DKB
1	1.0	1.0	1.0	0.7
2	1.0	0.95	0.975	0.685
3	2.0	1.852	0.951	0.67
4	3.0	2.709	0.927	0.654
5	4.0	3.524	0.905	0.638
6	5.0	4.298	0.883	0.622
7	6.0	5.033	0.862	0.605
8	7.0	5.732	0.841	0.588
9	8.0	6.395	0.822	0.57
10	9.0	7.025	0.803	0.553
11	10.0	7.624	0.784	0.535
12	11.0	8.192	0.766	0.517
13	12.0	8.733	0.749	0.498
14	13.0	9.246	0.732	0.479
15	14.0	9.734	0.716	0.462
16	15.0	10.19	0.7	0.445
17	0.0625	0.094	0.971	0.43
18	0.0625	0.094	0.971	0.451
19	0.125	0.193	0.943	0.473
20	0.1875	0.297	0.915	0.495
21	0.25	0.406	0.889	0.517
22	0.333	0.544	0.863	0.539
23	0.428	0.703	0.839	0.559
24	0.538	0.887	0.815	0.577
25	0.666	1.103	0.792	0.595
26	0.818	1.36	0.77	0.611
27	1.0	1.67	0.749	0.626
28	1.222	2.05	0.729	0.64
29	1.5	2.53	0.709	0.653
30	0.538	0.318	0.857	0.665
31	0.538	0.387	0.902	0.651
32	0.538	0.46	0.949	0.635
33	0.538	0.537	0.999	0.618
34	0.538	0.618	0.951	0.598
35	0.538	0.703	0.903	0.579
36	0.538	0.793	0.858	0.56
37	0.538	0.887	0.815	0.542
38	0.666	1.103	0.792	0.523
39	0.818	1.36	0.77	0.504
40	1.0	1.67	0.749	0.485
41	1.22	2.05	0.729	0.467
42	1.5	2.527	0.709	0.45
43	1.857	3.142	0.69	0.434
44	2.333	3.965	0.671	0.419
45	3.0	5.121	0.653	0.405
46	4.0	6.857	0.636	0.392
47	0.176	0.102	0.937	0.379
48	0.176	0.1605	0.986	0.397
49	0.176	0.222	0.963	0.416
50	0.176	0.286	0.915	0.436
51	0.25	0.407	0.889	0.456
52	0.333	0.544	0.863	0.477
53	0.428	0.703	0.839	0.497
54	0.538	0.887	0.815	0.518
55	0.666	1.104	0.792	0.538
56	0.818	1.36	0.77	0.556
57	1.0	1.67	0.749	0.573
58	1.222	2.05	0.729	0.589
59	1.5	2.527	0.709	0.604
60	1.857	3.142	0.69	0.618
61	2.333	3.965	0.671	0.631
62	3.0	5.121	0.653	0.644
63	4.0	6.857	0.636	0.655
64	5.666	9.757	0.62	0.666

Number of change	p value	q value	value of f	credibility weight of DKB
65	9.0	15.56	0.604	0.677
66	19.0	33.0	0.588	0.686
67	20.0	12.18	0.628	0.696
68	20.0	12.18	0.628	0.705
69	20.0	12.18	0.628	0.714
70	20.0	12.18	0.628	0.723
71	20.0	12.18	0.628	0.732
72	20.0	12.18	0.628	0.741
73	20.0	12.18	0.628	0.749
74	20.0	12.18	0.628	0.757
75	20.0	12.18	0.628	0.764
76	20.0	12.18	0.628	0.772
77	20.0	12.18	0.628	0.779
78	20.0	12.18	0.628	0.786
79	20.0	12.18	0.628	0.792
80	20.0	12.18	0.628	0.79

APPENDIX B

Example set for rule generation by inductive learning (see Section 6.5).

Table 1 shows values (1 to 59) for the 7 fuzzy premise variables:

fvar1 = substrate peak broadening;
fvar2 = substrate asymmetry in peak;
fvar3 = layer asymmetry in peak;
fvar4 = layer wedge shaped peak;
fvar5 = interference fringes;
fvar5 = visibility of interference fringes;
fvar7 = intensity of layer peak.

Table 1:

No	fvar1	fvar2	fvar3	fvar4	fvar5	fvar6	fvar7
1	0.15	0.05	0.0	0.05	0.05	0.5	0.8
2	0.15	0.05	0.0	0.1	0.05	0.8	0.8
3	0.15	0.05	0.0	0.1	0.05	0.8	0.8
4	0.15	0.05	0.0	0.1	0.05	0.5	0.8
5	0.15	0.05	0.0	0.05	0.25	0.8	0.8
6	0.15	0.05	0.0.	0.05	0.05	0.8	0.8
7	0.15	0.05	0.0	0.05	0.05	0.8	0.8
8	0.15	0.05	0.0	0.05	0.05	0.5	0.8
9	0.15	0.05	0.0	0.05	0.05	0.8	0.8
10	0.15	0.05	0.0	0.05	0.05	0.5	0.8
11	0.15	0.05	0.0	0.05	0.05	0.5	0.8
12	0.15	0.05	0.0	0.05	0.05	0.5	0.8
13	0.15	0.05	0.0	0.05	0.05	0.5	0.8
14	0.15	0.05	0.0	0.05	0.05	0.5	0.8
15	0.15	0.05	0.0	0.1	0.05	0.5	0.8
16	0.15	0.05	0.0	0.1	0.05	0.5	0.8
17	0.15	0.05	0.0	0.1	0.05	0.5	0.8
18	0.15	0.05	0.0	0.1	0.05	0.5	0.8
19	0.15	0.05	0.0	0.3	0.05	0.8	0.8
20	0.15	0.05	0.0	0.3	0.05	0.5	0.8
21	0.15	0.05	0.0	0.05	0.05	0.5	0.8
22	0.15	0.05	0.0	0.05	0.05	0.5	0.05
23	0.15	0.05	0.0	0.05	0.05	0.8	0.05
24	0.15	0.05	0.0	0.05	0.05	0.8	0.5
25	0.15	0.05	0.0	0.1	0.05	0.8	0.8
26	0.15	0.05	0.0	0.05	0.1	0.8	0.8
27	0.15	0.05	0.0	0.05	0.05	0.8	0.8
28	0.15	0.05	0.0	0.05	0.05	0.8	0.8
29	0.15	0.05	0.0	0.05	0.05	0.8	0.8
30	0.15	0.05	0.0	0.1	0.05	0.25	0.8
31	0.15	0.05	0.0	0.1	0.05	0.8	0.5
32	0.15	0.05	0.0	0.05	0.05	0.8	0.8
33	0.15	0.05	0.0	0.05	0.05	0.8	0.8
34	0.15	0.05	0.0	0.05	0.05	0.8	0.8
35	0.15	0.05	0.0	0.05	0.05	0.8	0.8
36	0.15	0.05	0.0	0.05	0.05	0.8	0.8
37	0.15	0.05	0.0	0.05	0.05	0.8	0.8
38	0.15	0.05	0.0	0.05	0.05	0.1	0.8
39	0.15	0.05	0.0	0.1	0.05	0.1	0.8
40	0.15	0.05	0.0	0.05	0.05	0.05	0.8
41	0.15	0.05	0.0	0.05	0.05	0.05	0.05
42	0.15	0.05	0.0	0.05	0.05	0.05	0.05
43	0.15	0.05	0.0	0.05	0.05	0.1	0.05
44	0.15	0.05	0.0	0.05	0.05	0.25	0.5
45	0.15	0.05	0.0	0.05	0.05	0.25	0.8
46	0.15	0.05	0.0	0.05	0.05	0.5	0.5
47	0.15	0.05	0.0	0.05	0.05	0.5	0.8
48	0.15	0.05	0.0	0.05	0.05	0.25	0.8
49	0.15	0.05	0.0	0.05	0.05	0.25	0.8
50	0.15	0.05	0.0	0.05	0.05	0.25	0.8
51	0.15	0.05	0.0	0.05	0.05	0.25	0.8
52	0.15	0.05	0.0	0.05	0.05	0.25	0.8
53	0.15	0.05	0.0	0.05	0.05	0.5	0.8
54	0.15	0.05	0.0	0.05	0.05	0.5	0.8
55	0.15	0.05	0.0	0.05	0.05	0.5	0.8
56	0.15	0.05	0.0	0.05	0.05	0.5	0.8
57	0.15	0.05	0.0	0.05	0.05	0.5	0.8
58	0.15	0.05	0.0	0.05	0.05	0.5	0.8
59	0.15	0.05	0.0	0.05	0.05	0.5	0.8

Table 2 shows the example set (1 to 59) of values for 16 Boolean premise variables:

b1 = type of structure is single layer;
b2 = number of peaks is one;
b3 = number of peaks is two;
b4 = number of peaks is more than two;
b5 = number of peaks is none;
b6 = substrate material equal to layer;
b7 = peak separation is low;
b8 = layer split peak;
b9 = layer integrated intensity of peak is zero;
b10 = layer thickness greater than half micron;
b11 = layer thickness less than 5 microns;
b12 = peak splitting is zero;
b13 = peak splitting less than three times width of peak;
b14 = relaxed mismatch is high;
b15 = peak splitting is high;
b16 = spacing of interference fringes is low.

Table 2:

No	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16
1	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
2	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
3	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
4	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
5	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
6	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
7	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
8	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
9	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
10	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
11	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
12	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
13	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
14	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
15	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
16	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
17	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
18	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
19	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
20	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
21	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1
22	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1
23	1	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0
24	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
25	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
26	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
27	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
28	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
29	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
30	1	0	1	0	0	0	0	1	0	1	1	0	0	0	1	0
31	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
32	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
33	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
34	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
35	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
36	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
37	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
38	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
39	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
40	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
41	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
42	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
43	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0
44	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
45	1	0	1	0	0	0	0	1	0	1	1	0	0	0	1	0
46	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
47	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
48	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
49	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
50	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
51	1	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0
52	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
53	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
54	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
55	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
56	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
57	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
58	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0
59	1	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0

Tables 3 and 4 show the example set (1 to 59) of values for the 18 fuzzy consequent variables:

fc1= crystal quality; fc2= characteristic curve;
fc3= misorientation of substrate; fc4= bending of substrate;
fc5= grading of the layer; fc6= layer is thick;
fc7= change in lattice parameter with depth; fc8 = layer is present in the substrate peak;
fc9= layer is thin; fc10= evidence of mismatch;
fc11= peak is outside the scan range; fc12= misoriented or mismatched layer;
fc13= multiple layers; fc14= simulation or calibration chart is needed;
fc15= relaxation; fc16= incorrect experiment;
fc17= thickness of layer; fc18= experimental mismatch;

Table 3:

No	fc1	fc2	fc3	fc4	fc5	fc6	fc7	fc8	fc9
1	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
2	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
3	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
4	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
5	0.35	0.85	0.125	0.125	0.35	0.43	0.0	0.1	0.35
6	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
7	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
8	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
9	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
10	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
11	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
12	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
13	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
14	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
15	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
16	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
17	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
18	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
19	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
20	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
21	0.35	0.85	0.125	0.125	0.1	0.23	0.6	0.1	0.0
22	0.35	0.85	0.125	0.125	0.1	0.23	0.6	0.1	0.0
23	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
24	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
25	0.35	0.85	0.125	0.125	0.25	0.43	0.0	0.1	0.35
26	0.35	0.85	0.125	0.125	0.24	0.43	0.0	0.1	0.35
27	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
28	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
29	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
30	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
31	0.35	0.85	0.125	0.125	0.23	0.43	0.0	0.1	0.35
32	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
33	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
34	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
35	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
36	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
37	0.35	0.85	0.125	0.125	0.1	0.43	0.0	0.1	0.35
38	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
39	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
40	0.35	0.85	0.125	0.125	0.1	0.15	0.0	0.1	0.35
41	0.35	0.85	0.125	0.125	0.1	0.23	0.6	0.1	0.0
42	0.35	0.85	0.125	0.125	0.1	0.23	0.0	0.1	0.0
43	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
44	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
45	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
46	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
47	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
48	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
49	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
50	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35

No	fc1	fc2	fc3	fc4	fc5	fc6	fc7	fc8	fc9
51	0.35	0.85	0.125	0.125	0.23	0.0	0.0	0.1	0.35
52	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
53	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
54	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
55	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
56	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
57	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
58	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35
59	0.35	0.85	0.125	0.125	0.1	0.0	0.0	0.1	0.35

Table 4:

No	fc10	fc11	fc12	fc13	fc14	fc15	fc16	fc17	fc18
1	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
2	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
3	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
4	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
6	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
7	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
8	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
9	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
10	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
11	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
15	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
16	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
17	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
18	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
19	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
20	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
21	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
22	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
23	0.275	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.85
24	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
27	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
28	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
29	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85
30	0.275	0.0	0.2	0.2	0.0	0.0	0.0	0.85	0.85
31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
32	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
33	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
34	0.35	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
35	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
36	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
37	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
38	0.275	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
39	0.275	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
40	0.275	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
41	0.275	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
42	0.35	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
43	0.275	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.85
44	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
45	0.275	0.0	0.2	0.2	0.0	0.0	0.0	0.85	0.85
46	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
47	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
48	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
49	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
51	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
52	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
53	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
54	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
55	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
56	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
57	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
58	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85
59	0.275	0.0	0.0	0.0	0.0	0.0	0.0	0.85	0.85

APPENDIX C

Complete fuzzy ruleset: split into 4 partitions.

CODES are representations of rule elements that are used to construct the rules.

PARTITION 1: RULES FOR SUBSTRATE ONLY STRUCTURE

		CODES
[1]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = EXTREME	F5
	THEN crystal_quality = NONE	C1,1
[2]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = VERY	F4
	THEN crystal_quality = SOME	C1,2
[3]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = FAIRLY	F3
	THEN crystal_quality = FAIRLY good	C1,3
[4]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = SOME	F2
	THEN crystal_quality = VERY good	C1,4
[5]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = NONE	F1
	THEN crystal_quality = EXTREME	C1,5
[6]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND substrate_asymmetry_in_peak = EXTREME	F10
	THEN strain_in_surface_layer_of_sample = EXTREME	C2,5
[7]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND substrate_asymmetry_in_peak = VERY	F9
	THEN strain_in_surface_layer_of_sample = VERY	C2,4
[8]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND substrate_asymmetry_in_peak = FAIRLY	F8
	THEN strain_in_surface_layer_of_sample = FAIRLY	C2,3
[9]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND substrate_asymmetry_in_peak = SOME	F7
	THEN strain_in_surface_layer_of_sample = SOME	C2,2
[10]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND substrate_asymmetry_in_peak = NONE	F6
	THEN strain_in_surface_layer_of_sample = NONE	C2,1
[11]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_more_than_one = TRUE	B5
	THEN reference_crystal_is_different_to_substrate = VERY	C3,4
[12]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_more_than_one = TRUE	B5
	THEN crystal_consists_of_subgrains = VERY	C4,4
[13]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	THEN characteristic_curve = EXTREME	C5,5
[14]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = EXTREME	F5
	THEN misorientation_of_substrate = EXTREME	C6,5
[15]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = VERY	F4
	THEN misorientation_of_substrate = VERY	C6,4
[16]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = FAIRLY	F3
	THEN misorientation_of_substrate = FAIRLY	C6,3
[17]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B2
	AND substrate_peak_broadening = SOME	F2
	THEN misorientation_of_substrate = SOME	C6,2
[18]	IF type_of_structure_is_substrate_only = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	AND substrate_peak_broadening = NONE	F1
	THEN misorientation_of_substrate = NONE	C6,1
[19]	IF number_of_peaks_is_none = TRUE	B7
	THEN incorrect_experiment = EXTREME	C7,4

PARTITION 2: RULES FOR SINGLE LAYER STRUCTURE

		CODES
[1]	IF type_of_structure_is_single_layer = TRUE	B1
	AND number_of_peaks_is_two = TRUE	B5
	THEN crystal_quality = FAIRLY good	C1,3
[2]	IF type_of_structure_is_single_layer = TRUE	B1
	AND substrate_peak_broadening = EXTREME	F5
	THEN bending_of_substrate = EXTREME	C4,5
[3]	IF type_of_structure_is_single_layer = TRUE	B1
	AND substrate_peak_broadening = VERY	F4
	THEN bending_of_substrate = VERY	C4,4
[4]	IF type_of_structure_is_single_layer = TRUE	B1
	AND substrate_peak_broadening = FAIRLY	F3
	THEN bending_of_substrate = FAIRLY	C4,3
[5]	IF type_of_structure_is_single_layer = TRUE	B1
	AND substrate_peak_broadening = SOME	F2
	THEN bending_of_substrate = SOME	C4,2

		CODES
[6]	IF type_of_structure_is_single_layer = TRUE	B1
	AND substrate_peak_broadening = NONE	F1
	THEN bending_of_substrate = NONE	C4,1
[7]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = EXTREME	F15
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = EXTREME	C5,5
[8]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = EXTREME	F15
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = EXTREME	C5,5
[9]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = EXTREME	F15
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN grading_of_the_layer = VERY	C5,4
[10]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = EXTREME	F15
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = FAIRLY	C5,3
[11]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = EXTREME	F15
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = FAIRLY	C5,3
[12]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = VERY	F14
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = EXTREME	C5,5
[13]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = VERY	F14
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = VERY	C5,4
[14]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = VERY	F14
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN grading_of_the_layer = VERY	C5,4
[15]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = VERY	F14
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = FAIRLY	C5,3
[16]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = VERY	F14
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = SOME	C5,2
[17]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = FAIRLY	F13
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = VERY	C5,4

		CODES
[18]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = FAIRLY	F13
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = VERY	C5,4
[19]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = FAIRLY	F13
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN grading_of_the_layer = FAIRLY	C5,3
[20]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = FAIRLY	F13
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = SOME	C5,2
[21]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = FAIRLY	F13
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = SOME	C5,2
[22]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = SOME	F12
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = FAIRLY	C5,1
[23]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = SOME	F12
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = FAIRLY	C5,3
[24]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = SOME	F12
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN grading_of_the_layer = SOME	C5,2
[25]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = SOME	F12
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = SOME	C5,2
[26]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = SOME	F12
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = NONE	C5,1
[27]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = NONE	F11
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = FAIRLY	C5,3
[28]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = NONE	F11
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = SOME	C5,2

		CODES
[29]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = NONE	F11
	AND layer_wedge_shaped_peak = FAIRLY	F17
	THEN grading_of_the_layer = SOME	C5,2
[30]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = NONE	F11
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = NONE	C5,1
[31]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND layer_asymmetry_in_peak = NONE	F11
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = NONE	C5,1
[32]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = EXTREME	F25
	AND visibility_of_interference_fringes = EXTREME	F30
	THEN layer_is_thick = VERY	C6,4
[33]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = EXTREME	F25
	AND visibility_of_interference_fringes = VERY	F29
	THEN layer_is_thick = VERY	C6,4
[34]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = EXTREME	F25
	AND visibility_of_interference_fringes = FAIRLY	F28
	THEN layer_is_thick = FAIRLY	C6,3
[35]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = EXTREME	F25
	AND visibility_of_interference_fringes = SOME	F27
	THEN layer_is_thick = SOME	C6,2
[36]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = EXTREME	F25
	AND visibility_of_interference_fringes = NONE	F26
	THEN layer_is_thick = SOME	C6,1
[37]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = VERY	F24
	AND visibility_of_interference_fringes = EXTREME	F30
	THEN layer_is_thick = VERY	C6,4
[38]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = VERY	F24
	AND visibility_of_interference_fringes = VERY	F29
	THEN layer_is_thick = VERY	C6,4
[39]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = VERY	F24
	AND visibility_of_interference_fringes = FAIRLY	F28
	THEN layer_is_thick = FAIRLY	C6,3

		CODES
[40]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = VERY	F24
	AND visibility_of_interference_fringes = SOME	F27
	THEN layer_is_thick = SOME	C6,2
[41]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = VERY	F24
	AND visibility_of_interference_fringes = NONE	F26
	THEN layer_is_thick = NONE	C6,1
[42]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = FAIRLY	F23
	AND visibility_of_interference_fringes = EXTREME	F30
	THEN layer_is_thick = FAIRLY	C6,3
[43]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = FAIRLY	F23
	AND visibility_of_interference_fringes = VERY	F29
	THEN layer_is_thick = FAIRLY	C6,3
[44]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = FAIRLY	F23
	AND visibility_of_interference_fringes = FAIRLY	F28
	THEN layer_is_thick = SOME	C6,2
[45]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = FAIRLY	F23
	AND visibility_of_interference_fringes = SOME	F27
	THEN layer_is_thick = SOME	C6,2
[46]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = FAIRLY	F23
	AND visibility_of_interference_fringes = NONE	F26
	THEN layer_is_thick = NONE	C6,1
[47]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = SOME	F22
	AND visibility_of_interference_fringes = EXTREME	F30
	THEN layer_is_thick = SOME	C6,2
[48]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = SOME	F22
	AND visibility_of_interference_fringes = VERY	F29
	THEN layer_is_thick = SOME	C6,2
[49]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = SOME	F22
	AND visibility_of_interference_fringes = FAIRLY	F28
	THEN layer_is_thick = SOME	C6,2
[50]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B13
	AND interference_fringes = SOME	F22
	AND visibility_of_interference_fringes = SOME	F27
	THEN layer_is_thick = NONE	C6,1

CODES

[51] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = SOME F22
AND visibility_of_interference_fringes = NONE F26

THEN layer_is_thick = NONE C6,1

[52] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = NONE F21
AND visibility_of_interference_fringes = EXTREME F30

THEN layer_is_thick = SOME C6,2

[53] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = NONE F21
AND visibility_of_interference_fringes = VERY F29

THEN layer_is_thick = NONE C6,1

[54] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = NONE F21
AND visibility_of_interference_fringes = FAIRLY F28

THEN layer_is_thick = NONE C6,1

[55] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = NONE F21
AND visibility_of_interference_fringes = SOME F27

THEN layer_is_thick = NONE C6,1

[56] IF type_of_structure_is_single_layer = TRUE B1
AND peak_separation_is_low = TRUE B13
AND interference_fringes = NONE F21
AND visibility_of_interference_fringes = NONE F26

THEN layer_is_thick = NONE C6,1

[57] IF type_of_structure_is_single_layer = TRUE B1
AND visibility_of_interference_fringes = NONE F26

THEN change_in_lattice_parameter_with_depth = VERY C7,4

[58] IF type_of_structure_is_single_layer = TRUE B1
AND substrate_asymmetry_in_peak = EXTREME F10

THEN layer_is_present_in_the_substrate_peak = FAIRLY C8,3

[59] IF type_of_structure_is_single_layer = TRUE B1
AND substrate_asymmetry_in_peak = VERY F9

THEN layer_is_present_in_the_substrate_peak = SOME C8,2

[60] IF type_of_structure_is_single_layer = TRUE B1
AND substrate_asymmetry_in_peak = FAIRLY F8

THEN layer_is_present_in_the_substrate_peak = SOME C8,2

[61] IF type_of_structure_is_single_layer = TRUE B1
AND substrate_asymmetry_in_peak = SOME F7

THEN layer_is_present_in_the_substrate_peak = NONE C8,1

[62] IF type_of_structure_is_single_layer = TRUE B1
AND substrate_asymmetry_in_peak = NONE F6

THEN layer_is_present_in_the_substrate_peak = NONE C8,1

[63] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_two = TRUE B5

THEN characteristic_curve = EXTREME C2,5

[64] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_one = TRUE B3

THEN layer_is_thin = SOME C9,2

CODES

[65] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_one = TRUE B3

THEN evidence_of_mismatch = SOME C10,2

[66] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_one = TRUE B3

THEN peak_is_outside_the_scan_range = SOME C11,2

[67] IF type_of_structure_is_single_layer = TRUE B1
AND layer_split_peak = TRUE B15

THEN misoriented_or_mismatched_layer = SOME C12,2

[68] IF type_of_structure_is_single_layer = TRUE B1
AND layer_split_peak = TRUE B15

THEN multiple_layers = SOME C13,2

[69] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_two = TRUE B5
AND layer_integrated_intensity_of_peak_is_zero = FALSE B18
AND layer_thickness_greater_than_half_a_micron = TRUE B19
AND layer_thickness_less_than_5_microns = TRUE B21

THEN thickness_of_layer = EXTREME C17,5
(Can calculate layer thickness from curve)

[70] IF type_of_structure_is_single_layer = TRUE B1
AND peak_splitting_is_zero = FALSE B24

THEN experimental_mismatch = EXTREME C18,5
(Can calculate experimental mismatch from curve)

[71] IF type_of_structure_is_single_layer = TRUE B1
AND number_of_peaks_is_two = TRUE B5
AND peak_splitting_less_than_
three_times_width_of_peak = TRUE B25

THEN simulation_or_calibration_
chart_is_needed = EXTREME C14,5

[72] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = EXTREME F35

THEN layer_is_thick = VERY C6,4

[73] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = VERY F34

THEN layer_is_thick = VERY C6,4

[74] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = FAIRLY F33

THEN layer_is_thick = VERY C6,4

[75] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = SOME F32

THEN layer_is_thick = FAIRLY C6,3

[76] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = NONE F31

THEN layer_is_thick = NONE C6,1

[77] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = NONE F31

THEN layer_is_thin = VERY C9,4

[78] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = SOME F32

THEN layer_is_thin = FAIRLY C9,3

[79] IF type_of_structure_is_single_layer = TRUE B1
AND intensity_of_layer_peak = FAIRLY F33

THEN layer_is_thin = SOME C9,2

		CODES
[80]	IF type_of_structure_is_single_layer = TRUE	B1
	AND intensity_of_layer_peak = VERY	F34
	THEN layer_is_thin = SOME	C9,2
[81]	IF type_of_structure_is_single_layer = TRUE	B1
	AND intensity_of_layer_peak = EXTREME	F35
	THEN layer_is_thin = NONE	C9,1
[82]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_asymmetry_in_peak = EXTREME	F15
	THEN evidence_of_mismatch = VERY	C10,5
[83]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_asymmetry_in_peak = VERY	F14
	THEN evidence_of_mismatch = VERY	C10,4
[84]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_asymmetry_in_peak = FAIRLY	F13
	THEN evidence_of_mismatch = VERY	C10,4
[85]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_asymmetry_in_peak = SOME	F12
	THEN evidence_of_mismatch = FAIRLY	C10,3
[86]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_asymmetry_in_peak = NONE	F11
	THEN evidence_of_mismatch = SOME	C10,2
[87]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B11
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN grading_of_the_layer = EXTREME	C5,5
[88]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B11
	AND layer_wedge_shaped_peak = VERY	F19
	THEN grading_of_the_layer = VERY	C5,4
[89]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B11
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN grading_of_the_layer = FAIRLY	C5,3
[90]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B11
	AND layer_wedge_shaped_peak = SOME	F17
	THEN grading_of_the_layer = SOME	C5,2
[91]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = TRUE	B11
	AND layer_wedge_shaped_peak = NONE	F16
	THEN grading_of_the_layer = NONE	C5,1
[92]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_wedge_shaped_peak = EXTREME	F20
	THEN evidence_of_mismatch = EXTREME	C10,5
[93]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_wedge_shaped_peak = VERY	F19
	THEN evidence_of_mismatch = VERY	C10,4

		CODES
[94]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_wedge_shaped_peak = FAIRLY	F18
	THEN evidence_of_mismatch = FAIRLY	C10,3
[95]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_wedge_shaped_peak = SOME	F17
	THEN evidence_of_mismatch = SOME	C10,2
[96]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_separation_is_low = FALSE	B12
	AND layer_wedge_shaped_peak = NONE	F16
	THEN evidence_of_mismatch = NONE	C10,1
[97]	IF evidence_of_mismatch = EXTREME	F40
	THEN relaxation = FAIRLY	C15,3
[98]	IF evidence_of_mismatch = VERY	F39
	THEN relaxation = SOME	C15,2
[99]	IF evidence_of_mismatch = FAIRLY	F38
	THEN relaxation = SOME	C15,2
[100]	IF evidence_of_mismatch = SOME	F37
	THEN relaxation = NONE	C15,1
[101]	IF evidence_of_mismatch = NONE	F36
	THEN relaxation = NONE	C15,1
[102]	IF type_of_structure_is_single_layer = TRUE	B1
	AND number_of_peaks_is_one = TRUE	B3
	THEN peak_is_outside_the_scan_range = FAIRLY	C11,3
[103]	IF type_of_structure_is_single_layer = TRUE	B1
	AND relaxed_mismatch_is_high = TRUE	B27
	THEN grading_of_the_layer = NONE	C5,1
[104]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_splitting_is_high = TRUE	B29
	THEN grading_of_the_layer = NONE	C5,1
[105]	IF type_of_structure_is_single_layer = TRUE	B1
	AND peak_splitting_is_high = TRUE	B29
	THEN evidence_of_mismatch = VERY	C10,4
[106]	IF type_of_structure_is_single_layer = TRUE	B1
	AND spacing_of_interference_fringes_is_low = TRUE	B31
	THEN layer_is_thick = FAIRLY	C6,3
[107]	IF type_of_structure_is_single_layer = TRUE	B1
	AND spacing_of_interference_fringes_is_low = FALSE	B32
	THEN layer_is_thin = FAIRLY	C9,3
[108]	IF type_of_structure_is_single_layer = TRUE	B1
	AND relaxed_mismatch_is_high = TRUE	B27
	THEN relaxation = FAIRLY	C15,3
[109]	IF type_of_structure_is_single_layer = TRUE	B1
	AND layer_thickness_greater_than_half_micron = FALSE	B20
	THEN simulation_or_calibration_chart_is_needed = VERY	C14
[110]	IF number_of_peaks_is_zero = TRUE	B9
	THEN incorrect_experiment = VERY	C16,4

		CODES
[111]	IF substrate_peak_broadening = EXTREME	F5
	THEN misorientation_of_substrate = VERY	C3,4
[112]	IF substrate_peak_broadening = VERY	F4
	THEN misorientation_of_substrate = VERY	C3,4
[113]	IF substrate_peak_broadening = FAIRLY	F3
	THEN misorientation_of_substrate = FAIRLY	C3,3
[114]	IF substrate_peak_broadening = SOME	F2
	THEN misorientation_of_substrate = SOME	C3,2
[115]	IF substrate_peak_broadening = NONE	F1
	THEN misorientation_of_substrate = NONE	C3,1
[116]	IF intensity_of_layer_peak = NONE	F31
	THEN misoriented_or_mismatched_layer = VERY	C12,4
[117]	IF intensity_of_layer_peak = SOME	F32
	THEN misoriented_or_mismatched_layer = FAIRLY	C12,3

PARTITION 3: RULES FOR MULTIPLE LAYERS STRUCTURE

		CODES
[1]	IF evidence_of_mismatch = EXTREME	F35
	THEN relaxation = FAIRLY	C4,3
[2]	IF evidence_of_mismatch = VERY	F34
	THEN relaxation = SOME	C4,2
[3]	IF evidence_of_mismatch = FAIRLY	F33
	THEN relaxation = SOME	C4,2
[4]	IF evidence_of_mismatch = SOME	F32
	THEN relaxation = NONE	C4,1
[5]	IF evidence_of_mismatch = NONE	F31
	THEN relaxation = NONE	C4,1
[6]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND correspondence_between_layers_and_peaks = TRUE	B3
	THEN layers_are_thick = VERY	C5,4
[7]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND correspondence_between_layers_and_peaks = FALSE	B3
	THEN there_are_hidden_layers_somewhere = VERY	C6,4
[8]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = EXTREME	F25
	THEN simulation_or_calibration_chart_is_needed = EXTREME	C3,5
[9]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = VERY	F24
	THEN simulation_or_calibration_chart_is_needed = VERY	C3,4

		CODES
[10]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = FAIRLY	F23
	THEN simulation_or_calibration_chart_is_needed = FAIRLY	C3,3
[11]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = SOME	F22
	THEN simulation_or_calibration_chart_is_needed = SOME	C3,2
[12]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = NONE	F21
	THEN simulation_or_calibration_chart_is_needed = NONE	C3,1
[13]	IF number_of_peaks_is_none = TRUE	B5
	THEN incorrect_experiment = VERY	C7,4
[14]	IF multiple_layers_wedge_shaped_peak = EXTREME	F16
	AND visibility_of_interference_fringes = NONE	F30
	THEN grading = EXTREME	C1,5
[15]	IF multiple_layers_wedge_shaped_peak = VERY	F17
	AND visibility_of_interference_fringes = SOME	F29
	THEN grading = VERY	C1,4
[16]	IF multiple_layers_wedge_shaped_peak = FAIRLY	F18
	AND visibility_of_interference_fringes = FAIRLY	F28
	THEN grading = FAIRLY	C1,3
[17]	IF multiple_layers_wedge_shaped_peak = SOME	F19
	AND visibility_of_interference_fringes = VERY	F27
	THEN grading = SOME	C1,2
[18]	IF multiple_layers_wedge_shaped_peak = NONE	F20
	AND visibility_of_interference_fringes = EXTREME	F26
	THEN grading = NONE	C1,1
[19]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = EXTREME	F25
	THEN there_are_hidden_layers_somewhere = EXTREME	C6,5
[20]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = VERY	F24
	THEN there_are_hidden_layers_somewhere = VERY	C6,4
[21]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = FAIRLY	F23
	THEN there_are_hidden_layers_somewhere = FAIRLY	C6,3
[22]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = SOME	F22
	THEN there_are_hidden_layers_somewhere = SOME	C6,2
[23]	IF type_of_structure_is_multiple_layers = TRUE	B1
	AND interference_fringes = NONE	F21
	THEN there_are_hidden_layers_somewhere = NONE	C6,1
[24]	IF interference_fringes = EXTREME	F25
	AND correspondence_between_layers_and_peaks = FALSE	B4
	THEN interferometer_structure = EXTREME	C10,5
[25]	IF interference_fringes = VERY	F24
	AND correspondence_between_layers_and_peaks = FALSE	B4
	THEN interferometer_structure = VERY	C10,4

		CODES
[26]	IF interference_fringes = FAIRLY AND correspondence_between_layers_and_peaks = FALSE	F23 B4
	THEN interferometer_structure = FAIRLY	C10,3
[27]	IF interference_fringes = SOME AND correspondence_between_layers_and_peaks = FALSE	F22 B4
	THEN interferometer_structure = SOME	C10,2
[28]	IF interference_fringes = NONE AND correspondence_between_layers_and_peaks = FALSE	F21 B4
	THEN interferometer_structure = NONE	C10,1
[29]	IF correspondence_between_layers_and_peaks = FALSE	B4
	THEN evidence_of_mismatch = FAIRLY	C8,3
[30]	IF correspondence_between_layers_and_peaks = FALSE	B4
	THEN relaxation = FAIRLY	C4,3
[31]	IF split_substrate_peak = TRUE	B7
	THEN there_are_thin_layers_at_the_interfaces = VERY	C9,4

PARTITION 4: RULES FOR MQW AND SUPERLATTICE STRUCTURES

		CODES
[1]	IF evidence_of_mismatch = EXTREME	F45
	THEN relaxation = FAIRLY	C5,3
[2]	IF evidence_of_mismatch = VERY	F44
	THEN relaxation = SOME	C5,2
[3]	IF evidence_of_mismatch = FAIRLY	F43
	THEN relaxation = SOME	C5,2
[4]	IF evidence_of_mismatch = SOME	F42
	THEN relaxation = NONE	C5,1
[5]	IF evidence_of_mismatch = NONE	F41
	THEN relaxation = NONE	C5,1
[6]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = EXTREME	B1 F15
	THEN layers_are_thin = EXTREME	C7,5
[7]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = VERY	B1 F14
	THEN layers_are_thin = VERY	C7,4
[8]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = FAIRLY	B1 F13
	THEN layers_are_thin = FAIRLY	C7,3
[9]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = SOME	B1 F12
	THEN layers_are_thin = SOME	C7,2

		CODES
[10]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = NONE	B1 F11
	THEN layers_are_thin = NONE	C7,1
[11]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = EXTREME	B1 F15
	THEN layers_are_thick = NONE	C6,1
[12]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = VERY	B1 F14
	THEN layers_are_thick = SOME	C6,2
[13]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = FAIRLY	B1 F13
	THEN layers_are_thick = FAIRLY	C6,3
[14]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = SOME	B1 F12
	THEN layers_are_thick = VERY	C6,4
[15]	IF type_of_structure_is_MQW = TRUE AND separation_of_satellite_peaks = NONE	B1 F11
	THEN layers_are_thick = EXTREME	C6,5
[16]	IF type_of_structure_is_MQW = TRUE AND satellite_spacing_greater_than_zero = TRUE	B1 B3
	THEN period_of_superlattice = EXTREME (Can calculate the period of the superlattice from the curve)	C13,5
[17]	IF type_of_structure_is_MQW = TRUE AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	B1 F10
	THEN grading_or_dispersion_of_ layer_thicknesses = EXTREME	C8,5
[18]	IF type_of_structure_is_MQW = TRUE AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	B1 F9
	THEN grading_or_dispersion_of_ layer_thicknesses = VERY	C8,4
[19]	IF type_of_structure_is_MQW = TRUE AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	B1 F8
	THEN grading_or_dispersion_of_ layer_thicknesses = FAIRLY	C8,3
[20]	IF type_of_structure_is_MQW = TRUE AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	B1 F7
	THEN grading_or_dispersion_of_ layer_thicknesses = SOME	C8,2
[21]	IF type_of_structure_is_MQW = TRUE AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	B1 F6
	THEN grading_or_dispersion_of_ layer_thicknesses = NONE	C8,1
[22]	IF type_of_structure_is_MQW = TRUE AND satellite_visibility = NONE	B1 F1
	THEN grading_occurs_through_AB_layers = EXTREME	C9,5
[23]	IF type_of_structure_is_MQW = TRUE AND satellite_visibility = SOME	B1 F2
	THEN grading_occurs_through_AB_layers = VERY	C9,4

		CODES
[24]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = FAIRLY	F3
	THEN grading_occurs_through_AB_layers = FAIRLY	C9,3
[25]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = VERY	F4
	THEN grading_occurs_through_AB_layers = SOME	C9,2
[26]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = EXTREME	F5
	THEN grading_occurs_through_AB_layers = NONE	C9,1
[27]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_subsidiary_interference_effects = TRUE	B5
	THEN large_overall_layer_thickness = VERY	C10,4
[28]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_broadening_of_higher_order_peaks = TRUE	B7
	THEN grading_or_dispersion_of_layer_thicknesses = VERY	C8,4
[29]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = EXTREME	F15
	AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	F10
	THEN characteristic_curve = NONE	C2,1
[30]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = EXTREME	F15
	AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	F9
	THEN characteristic_curve = NONE	C6,1
[31]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = EXTREME	F15
	AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	F8
	THEN characteristic_curve = FAIRLY	C2,3
[32]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = EXTREME	F15
	AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	F7
	THEN characteristic_curve = EXTREME	C2,5
[33]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = EXTREME	F15
	AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	F6
	THEN characteristic_curve = EXTREME	C2,5
[34]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = VERY	F14
	AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	F10
	THEN characteristic_curve = NONE	C2,1
[35]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = VERY	F14
	AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	F9
	THEN characteristic_curve = NONE	C2,1
[36]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = VERY	F14
	AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	F8
	THEN characteristic_curve = FAIRLY	C2,3
[37]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = VERY	F14
	AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	F7
	THEN characteristic_curve = EXTREME	C2,5
[38]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = VERY	F14
	AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	F6
	THEN characteristic_curve = EXTREME	C2,5
[39]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = FAIRLY	F13
	AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	F10
	THEN characteristic_curve = NONE	C2,1
[40]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = FAIRLY	F13
	AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	F9
	THEN characteristic_curve = NONE	C2,1
[41]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = FAIRLY	F13
	AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	F8
	THEN characteristic_curve = FAIRLY	C2,3
[42]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = FAIRLY	F13
	AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	F7
	THEN characteristic_curve = FAIRLY	C2,3
[43]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = FAIRLY	F13
	AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	F6
	THEN characteristic_curve = FAIRLY	C2,3
[44]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = SOME	F12
	AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	F10
	THEN characteristic_curve = NONE	C2,1
[45]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = SOME	F12
	AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	F9
	THEN characteristic_curve = NONE	C2,1
[46]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = SOME	F12
	AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	F8
	THEN characteristic_curve = NONE	C2,1
[47]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = SOME	F12
	AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	F7
	THEN characteristic_curve = SOME	C2,2
[48]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = SOME	F12
	AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	F6
	THEN characteristic_curve = SOME	C2,2

		CODES
[49]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = NONE	F11
	AND satellite_asymmetry_of_ plus_and_minus_peaks = EXTREME	F10
	THEN characteristic_curve = NONE	C2,1
[50]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = NONE	F11
	AND satellite_asymmetry_of_ plus_and_minus_peaks = VERY	F9
	THEN characteristic_curve = NONE	C2,1
[51]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = NONE	F11
	AND satellite_asymmetry_of_ plus_and_minus_peaks = FAIRLY	F8
	THEN characteristic_curve = NONE	C2,1
[52]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = NONE	F11
	AND satellite_asymmetry_of_ plus_and_minus_peaks = SOME	F7
	THEN characteristic_curve = NONE	C2,1
[53]	IF type_of_structure_is_MQW = TRUE	B1
	AND separation_of_satellite_peaks = NONE	F11
	AND satellite_asymmetry_of_ plus_and_minus_peaks = NONE	F6
	THEN characteristic_curve = NONE	C2,1
[54]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = NONE	F1
	THEN layers_are_not_uniform = EXTREME	C11,5
[55]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = SOME	F2
	THEN layers_are_not_uniform = FAIRLY	C11,3
[56]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = FAIRLY	F3
	THEN layers_are_not_uniform = SOME	C11,2
[57]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = VERY	F4
	THEN layers_are_not_uniform = NONE	C11,1
[58]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_visibility = EXTREME	F5
	THEN layers_are_not_uniform = NONE	C11,1
[59]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_relative_width_of_ peaks_greater_than_zero = FALSE	B10
	THEN crystal_quality = NONE	C1,1
[60]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_relative_width_of_ peaks_greater_than_zero = TRUE	B11
	THEN period_of_superlattice = EXTREME	C13,5
	(Can calculate the period of the superlattice from the curve)	
[61]	IF type_of_structure_is_MQW = TRUE	B1
	AND thickness_of_superlattice_ less_than_half_micron = TRUE	B13
	THEN simulation_or_calibration_ chart_is_needed = EXTREME	C4,5

		CODES
[62]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_relative_integrated_ intensities_greater_than_zero = FALSE	B11
	THEN crystal_quality = NONE	C1,1
[63]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_relative_width_of_ peaks_greater_than_zero = TRUE	B9
	THEN period_dispersion = EXTREME	C14,5
	(Can calculate period dispersion from curve)	
[64]	IF type_of_structure_is_MQW = TRUE	B1
	AND satellite_relative_width_of_ peaks_greater_than_zero = TRUE	B11
	THEN crystal_quality = FAIRLY	C1,3
[65]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND interference_fringes = EXTREME	F35
	THEN layers_are_thin = EXTREME	C7,5
[66]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND interference_fringes = VERY	F34
	THEN layers_are_thin = VERY	C7,4
[67]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND interference_fringes = FAIRLY	F33
	THEN layers_are_thin = FAIRLY	C7,3
[68]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND interference_fringes = SOME	F32
	THEN layers_are_thin = SOME	C7,2
[69]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND interference_fringes = NONE	F31
	THEN layers_are_thin = NONE	C7,1
[70]	IF type_of_structure_is_MQW = TRUE	B1
	AND type_of_structure_has_additional_layers = TRUE	B15
	AND satellite_broadening_of_higher_order_peaks = TRUE	B7
	THEN grading_or_dispersion_of_layer_thicknesses = VERY	C8,4
[71]	IF number_of_peaks_is_none = TRUE	B1
	THEN incorrect_experiment = VERY	C12,4
[72]	IF substrate_peak_broadening = EXTREME	F5
	THEN misorientation_of_substrate = VERY	C3,4
[73]	IF substrate_peak_broadening = VERY	F4
	THEN misorientation_of_substrate = VERY	C3,4
[74]	IF substrate_peak_broadening = FAIRLY	F3
	THEN misorientation_of_substrate = FAIRLY	C3,3
[75]	IF substrate_peak_broadening = SOME	F2
	THEN misorientation_of_substrate = SOME	C3,2
[76]	IF substrate_peak_broadening = NONE	F1
	THEN misorientation_of_substrate = NONE	C3,1

APPENDIX D:

Complete listing of the POP-11 program for the fuzzy system for X-ray rocking curve analysis.

Program Files:

[1]	initial.p;	[5]	rules2.p;	[9]	fuzzout.p;
[2]	fuzzy.p;	[6]	rules3.p;	[10]	thick.p;
[3]	setfuzz.p;	[7]	rules4.p;	[11]	induce.p;
[4]	rules1.p;	[8]	rules.p;	[12]	editrule.p

[1] The program 'initial.p' initialises the main variables in the fuzzy system.

```
;;; Program file: "initial.p"
;;;
;;; This program initializes the system

;;; 1. Function to add rules to a connection matrix
;;; 2. Connection-Matrix Object
;;; 3. Fuzzy-System Object


;;; [1] FUNCTION TO ADD RULES TO A CONNECTION MATRIX
;;;     (This function uses recursion)

define add_rules_to_CM(matrix,list_of_rules);
 lvars rule_number row column;

 if list_of_rules=[]
 then
 else
  hd(list_of_rules) -> rule_number;
  (rule_number * 2)-1 -> row;
  (rule_number * 2) -> column;

  1 -> matrix(row,column);
  add_rules_to_CM(matrix,tl(list_of_rules))
 endif;
enddefine;

;;; [2] CONNECTION_MATRIX OBJECT

flavour CONNECTION_MATRIX;
 ivars matrix_name matrix list_of_rules credibility_weight history_of_decisions
history_of_incs_decs;

 ;;; LIST OF METHODS
 ;;;
 ;;; 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; 2) save_to_file(x);

 ;;; METHOD: 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; Method to create a connection matrix
 defmethod create_a_connection_matrix(number_of_rules rules);
  lvars number_of_rules rules size_of_matrix;

  rules -> list_of_rules;

  number_of_rules * 2 -> size_of_matrix;

  newarray([1 ^size_of_matrix 1 ^size_of_matrix],0) -> matrix;

  ;;; Add rules to the Connection Matrix (uses recursion)
  add_rules_to_CM(matrix,rules)
 enddefmethod;

 ;;; METHOD 2: save connection matrices to file
 defmethod save_to_file(expert, structure);
  lvars expert structure filed;

  if expert = 1
  then
   if structure = 1
   then
    syscreate("me1s1",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   elseif structure = 2
   then
    syscreate("me1s2",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   elseif structure = 3
   then
    syscreate("me1s3",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   else
    syscreate("me1s4",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   endif;
  elseif expert = 2
  then
   if structure = 1
   then
    syscreate("me2s1",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   elseif structure = 2
   then
    syscreate("me2s2",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   elseif structure = 3
   then
    syscreate("me2s3",1,"line") -> filed;
    matrix -> datafile(filed);
    sysclose(filed);
   else
    syscreate("me2s4",1,"line") -> filed;
```

```
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  elseif expert = 3
  then
    if structure = 1
    then
      syscreate("me3s1",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 2
    then
      syscreate("me3s2",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 3
    then
      syscreate("me3s3",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    else
      syscreate("me3s4",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  elseif expert = 4
  then
    if structure = 1
    then
      syscreate("me4s1",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 2
    then
      syscreate("me4s2",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 3
    then
      syscreate("me4s3",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    else
      syscreate("me4s4",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  else
    if structure = 1
    then
      syscreate("me5s1",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 2
    then
      syscreate("me5s2",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 3
    then
      syscreate("me5s3",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    else
      syscreate("me5s4",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  endif;
enddefmethod;

endflavour;


;;; [3] FUZZY_SYSTEM OBJECT

;;; LIST OF METHODS:
;;;
;;; 1) initialize_the_system_variables;
;;;
```

```
flavour FUZZY_SYSTEM;
  ivars number_of_expert;

  defmethod initialize_the_system_variables;
    lvars fuzzy_peak_broadening fuzzy_asymmetry fuzzy_wedge_shaped;
    lvars fuzzy_visibility fuzzy_spacing_of_peaks fuzzy_plus_minus_asymmetry;
    lvars fuzzy_interference fuzzy_evidence_of_mismatch fuzzy_crystal_quality;
    lvars fuzzy_grading_of_the_layer fuzzy_layer_is_thick fuzzy_layer_is_thin;

    lvars number_of_rules existing_rules number_of_fuzzy_premises;
    lvars number_of_boolean_premises;
    lvars number_of_consequences fuzzy_premises boolean_premises;
    lvars consequences rule_variable;

    lvars filed i layers;

    lvars number_of_experts connection_matrix rules_for_substrate_only;
    lvars rules_for_single_layer rules_for_multiple_layers rules_for_MQW;
    lvars cred_weight hist_of_decisions hist_of_incs_decs;

    lvars percent_for_success percent_for_failure N ro kappa;

    lvars angst rule_history masking_of_rule number_of_fuzzy_variables;
    lvars membership_functions;
    lvars variable_history var_changes fuzzy_rules rule_variable mf;

    lvars history_of_real_decision_values array_of_lists p_values q_values;
    lvars f_values inc_values;

    lvars positive_threshold consequent_tally example_set_fuzzy_premise;
    lvars example_set_boolean_premise;
    lvars example_set_consequent consequents_in_rules fuzzy_intensity_of_peak;
    lvars potential_rules number_of_consequents;

    ;;; This method initialises system variables and stores
    ;;; these values to a series of files.

    nl(10);
    pr('THIS PROGRAM INITIALIZES SYSTEM VARIABLES IN A FUZZY
            SYSTEM FOR');nl(2);
    pr('          X-RAY ROCKING CURVE ANALYSIS');nl(5);

    ;;; FUZZY MEMBERSHIP FUNCTIONS
    [0.0 0.15 0.1 0.2 0.15 0.5 0.4 0.8 0.7 1.0] -> fuzzy_peak_broadening;
    [0.0 0.15 0.1 0.2 0.15 0.5 0.4 0.8 0.7 1.0] -> fuzzy_asymmetry;
    [0.0 0.15 0.1 0.2 0.15 0.4 0.3 0.7 0.6 1.0] -> fuzzy_wedge_shaped;
    [0.0 0.15 0.1 0.2 0.15 0.4 0.3 0.7 0.6 1.0] -> fuzzy_visibility;
    [0.0 0.15 0.1 0.3 0.2 0.5 0.4 0.7 0.6 1.0] -> fuzzy_spacing_of_peaks;
[0.0 0.15 0.1 0.2 0.15 0.5 0.4 0.8 0.7 1.0] -> fuzzy_plus_minus_asymmetry;
    [0.0 0.15 0.1 0.2 0.15 0.4 0.3 0.7 0.6 1.0] -> fuzzy_interference;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_evidence_of_mismatch;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_crystal_quality;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_grading_of_the_layer;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_layer_is_thick;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_layer_is_thin;
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_intensity_of_peak;

    ;;; VARIABLES DESCRIBING THE RULESETS
    pr('RECORD THE FUZZY RULESETS');nl(2);

    ;;;_____ ____ _____

    ;;; Substrate Only Structure
    ;;;_____

    pr('Substrate Only Structure');nl(2);

    19 -> number_of_rules;
    syscreate("rno1",1,"line") ->filed;
    number_of_rules -> datafile(filed);
    sysclose(filed);
    ;;; Record the ruleset in a matrix
    ;;;
    ;;; Column 1: fuzzy premises
    ;;; Column 2: boolean premises
    ;;; Column 3: conclusions
    ;;; Column 4: Number of conclusion
    pr('Record the ruleset');nl(1);
    newarray([1 ^number_of_rules 1 4]) -> existing_rules;
    ;;; Rule 1
    [5] -> existing_rules(1,1);
    [1 3] -> existing_rules(1,2);
    [1] -> existing_rules(1,3);
    [1] -> existing_rules(1,4);
```

```
;;; Rule 2
[4] -> existing_rules(2,1);
[1 3] -> existing_rules(2,2);
[2] -> existing_rules(2,3);
[1] -> existing_rules(2,4);
;;; Rule 3
[3] -> existing_rules(3,1);
[1 3] -> existing_rules(3,2);
[3] -> existing_rules(3,3);
[1] -> existing_rules(3,4);
;;; Rule 4
[2] -> existing_rules(4,1);
[1 3] -> existing_rules(4,2);
[4] -> existing_rules(4,3);
[1] -> existing_rules(4,4);
;;; Rule 5
[1] -> existing_rules(5,1);
[1 3] -> existing_rules(5,2);
[5] -> existing_rules(5,3);
[1] -> existing_rules(5,4);
;;; Rule 6
[10] -> existing_rules(6,1);
[1] -> existing_rules(6,2);
[5] -> existing_rules(6,3);
[2] -> existing_rules(6,4);
;;; Rule 7
[9] -> existing_rules(7,1);
[1] -> existing_rules(7,2);
[4] -> existing_rules(7,3);
[2] -> existing_rules(7,4);
;;; Rule 8
[8] -> existing_rules(8,1);
[1] -> existing_rules(8,2);
[3] -> existing_rules(8,3);
[2] -> existing_rules(8,4);
;;; Rule 9
[7] -> existing_rules(9,1);
[1] -> existing_rules(9,2);
[2] -> existing_rules(9,3);
[2] -> existing_rules(9,4);
;;; Rule 10
[6] -> existing_rules(10,1);
[1] -> existing_rules(10,2);
[1] -> existing_rules(10,3);
[2] -> existing_rules(10,4);
;;; Rule 11
[] -> existing_rules(11,1);
[1 5] -> existing_rules(11,2);
[4] -> existing_rules(11,3);
[3] -> existing_rules(11,4);
;;; Rule 12
[] -> existing_rules(12,1);
[1 5] -> existing_rules(12,2);
[4] -> existing_rules(12,3);
[4] -> existing_rules(12,4);
;;; Rule 13
[] -> existing_rules(13,1);
[1 3] -> existing_rules(13,2);
[5] -> existing_rules(13,3);
[5] -> existing_rules(13,4);
;;; Rule 14
[5] -> existing_rules(14,1);
[1 3] -> existing_rules(14,2);
[5] -> existing_rules(14,3);
[6] -> existing_rules(14,4);
;;; Rule 15
[4] -> existing_rules(15,1);
[1 3] -> existing_rules(15,2);
[4] -> existing_rules(15,3);
[6] -> existing_rules(15,4);
;;; Rule 16
[3] -> existing_rules(16,1);
[1 3] -> existing_rules(16,2);
[3] -> existing_rules(16,3);
[6] -> existing_rules(16,4);
;;; Rule 17
[2] -> existing_rules(17,1);
[1 3] -> existing_rules(17,2);
[2] -> existing_rules(17,3);
[6] -> existing_rules(17,4);
;;; Rule 18
[1] -> existing_rules(18,1);
[1 3] -> existing_rules(18,2);
[2] -> existing_rules(18,3);
[6] -> existing_rules(18,4);
;;; Rule 19
[] -> existing_rules(19,1);
[7] -> existing_rules(19,2);
[4] -> existing_rules(19,3);
[7] -> existing_rules(19,4);
syscreate("rules1",1,"line") -> filed;
existing_rules -> datafile(filed);
sysclose(filed);

4 -> number_of_fuzzy_premises;
syscreate("noffp1",1,"line") ->filed;
number_of_fuzzy_premises -> datafile(filed);
sysclose(filed);
4 -> number_of_boolean_premises;
syscreate("nofbp1",1,"line") ->filed;
number_of_boolean_premises -> datafile(filed);
sysclose(filed);
7 -> number_of_consequences;
syscreate("noffc1",1,"line") ->filed;
number_of_consequences -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
"substrate_peak_broadening" -> fuzzy_premises(1);
"substrate_asymmetry_in_peak" -> fuzzy_premises(2);
"interference_fringes" -> fuzzy_premises(3);
"visibility_of_interference_fringes" -> fuzzy_premises(4);
syscreate("fp1",1,"line") ->filed;
fuzzy_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
"type_of_structure_is_substrate_only" -> boolean_premises(1);
"number_of_peaks_is_one" -> boolean_premises(2);
"number_of_peaks_is_more_than_one" -> boolean_premises(3);
"number_of_peaks_is_none" -> boolean_premises(4);
syscreate("bp1",1,"line") ->filed;
boolean_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_consequences]) -> consequences;
"crystal_quality" -> consequences(1);
"strain_in_surface_layer_of_sample" -> consequences(2);
"reference_crystal_is_different_to_substrate" -> consequences(3);
"crystal_consists_of_subgrains" -> consequences(4);
"characteristic_curve" -> consequences(5);
"misorientation_of_substrate" -> consequences(6);
"incorrect_experiment" -> consequences(7);
syscreate("fc1",1,"line") ->filed;
consequences -> datafile(filed);
sysclose(filed);

1 -> angst;
syscreate("angst1",1,"line") -> filed;
angst -> datafile(filed);
sysclose(filed);

number_of_consequences * 5 -> number_of_consequents;
newarray([1 ^number_of_consequents 1 4]) -> potential_rules;
1 -> i;
until i > number_of_consequents
do
  [] -> potential_rules(i,1);
  [] -> potential_rules(i,2);
  [] -> potential_rules(i,3);
  [] -> potential_rules(i,4);
  i+1 -> i
enduntil;
syscreate("potrule1",1,"line") -> filed;
potential_rules -> datafile(filed);
sysclose(filed);

pr('Record the rule histories and maskings for rules');nl(1);
;;;  Array of lists containing the history of decisions for each rule
newarray([1 ^number_of_rules]) -> rule_history;
;;;  Array used to record if a rule is ASSERTed or UNASSERTed.
newarray([1 ^number_of_rules]) -> masking_of_rule;
1 -> i;
until i > number_of_rules
do
  [] -> rule_history(i);
  "ASSERT" -> masking_of_rule(i);
  i+1 -> i
```

```
enduntil;
syscreate("rhist1",1,"line") -> filed;
rule_history -> datafile(filed);
sysclose(filed);
syscreate("rmask1",1,"line") -> filed;
masking_of_rule -> datafile(filed);
sysclose(filed);

15 -> number_of_fuzzy_variables;
syscreate("fuzzno1",1,"line") -> filed;
number_of_fuzzy_variables -> datafile(filed);
sysclose(filed);

pr('Record the membership functions');nl(1);
;;; MEMBERSHIP FUNCTIONS ARE SEPARATED AND STORED IN
        AN ARRAY OF LISTS
newarray([1 ^number_of_fuzzy_premises]) -> membership_functions;
fuzzy_peak_broadening -> membership_functions(1);
fuzzy_asymmetry -> membership_functions(2);
fuzzy_interference -> membership_functions(3);
fuzzy_visibility -> membership_functions(4);
syscreate("memfunc1",1,"line") -> filed;
membership_functions -> datafile(filed);
sysclose(filed);

;;; The history of values is stored for each fuzzy variable
;;; two lists are used to store values that were successful
;;; and values which were unsuccessful
pr('Record the variable histories');nl(3);
newarray([1 2 1 ^number_of_fuzzy_variables]) -> variable_history;
newarray([1 ^number_of_fuzzy_variables]) -> var_changes;
1 -> i;
until i > number_of_fuzzy_variables
do
  [] -> variable_history(1,i);
  [] -> variable_history(2,i);
  [] -> var_changes(i);
  i+1 -> i
enduntil;
syscreate("varhist1",1,"line") -> filed;
variable_history -> datafile(filed);
sysclose(filed);
syscreate("varch1",1,"line") -> filed;
var_changes -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises 1 2]) -> fuzzy_rules;
[1 2 3 4 5 6 7 8 9 10 14 15 16 17 18] -> fuzzy_rules(1,1);
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] -> fuzzy_rules(1,2);
2 -> i;
until i > number_of_fuzzy_premises
do
  [] -> fuzzy_rules(i,1);
  [] -> fuzzy_rules(i,2);
  i+1 -> i
enduntil;
syscreate("frules1",1,"line") -> filed;
fuzzy_rules -> datafile(filed);
sysclose(filed);

;;; MEMBERSHIP FUNCTION USED BY EACH FUZZY VARIABLE
newarray([1 ^number_of_fuzzy_variables]) -> mf;
1 -> mf(1);
1 -> mf(2);
1 -> mf(3);
1 -> mf(4);
1 -> mf(5);
2 -> mf(6);
2 -> mf(7);
2 -> mf(8);
2 -> mf(9);
2 -> mf(10);
1 -> mf(11);
1 -> mf(12);
1 -> mf(13);
1 -> mf(14);
1 -> mf(15);
syscreate("memf1",1,"line") -> filed;
mf -> datafile(filed);
sysclose(filed);

[2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1] -> layers;
syscreate("l1",1,"line") -> filed;
layers -> datafile(filed);
sysclose(filed);

;;; Linguistic variables used in fuzzy rules
newarray([1 ^number_of_fuzzy_variables]) -> rule_variable;
'EXTREME' -> rule_variable(1);
'VERY' -> rule_variable(2);
'FAIRLY' -> rule_variable(3);
'SOME' -> rule_variable(4);
'NONE' -> rule_variable(5);
'EXTREME' -> rule_variable(6);
'VERY' -> rule_variable(7);
'FAIRLY' -> rule_variable(8);
'SOME' -> rule_variable(9);
'NONE' -> rule_variable(10);
'EXTREME' -> rule_variable(11);
'VERY' -> rule_variable(12);
'FAIRLY' -> rule_variable(13);
'SOME' -> rule_variable(14);
'NONE' -> rule_variable(15);
syscreate("rvar1",1,"line") -> filed;
rule_variable -> datafile(filed);
sysclose(filed);


;;; ____________________________________________________
;;;
;;; Single Layer Structure
;;; ____________________________________________________

pr('Single Layer Structure');nl(2);

117 -> number_of_rules;
syscreate("rno2",1,"line") ->filed;
number_of_rules -> datafile(filed);
sysclose(filed);
;;; Record the ruleset in a matrix
;;;
;;; Column 1: fuzzy premises
;;; Column 2: boolean premises
;;; Column 3: conclusions
;;; Column 4: Number of conclusion
pr('Record the ruleset');nl(1);
newarray([1 ^number_of_rules 1 4]) -> existing_rules;
;;; Rule 1
[] -> existing_rules(1,1);
[1 5] -> existing_rules(1,2);
[3] -> existing_rules(1,3);
[1] -> existing_rules(1,4);
;;; Rule 2
[5] -> existing_rules(2,1);
[1] -> existing_rules(2,2);
[5] -> existing_rules(2,3);
[4] -> existing_rules(2,4);
;;; Rule 3
[4] -> existing_rules(3,1);
[1] -> existing_rules(3,2);
[4] -> existing_rules(3,3);
[4] -> existing_rules(3,4);
;;; Rule 4
[3] -> existing_rules(4,1);
[1] -> existing_rules(4,2);
[3] -> existing_rules(4,3);
[4] -> existing_rules(4,4);
;;; Rule 5
[2] -> existing_rules(5,1);
[1] -> existing_rules(5,2);
[2] -> existing_rules(5,3);
[4] -> existing_rules(5,4);
;;; Rule 6
[1] -> existing_rules(6,1);
[1] -> existing_rules(6,2);
[1] -> existing_rules(6,3);
[4] -> existing_rules(6,4);
;;; Rule 7
[15 20] -> existing_rules(7,1);
[1 13] -> existing_rules(7,2);
[5] -> existing_rules(7,3);
[5] -> existing_rules(7,4);
;;; Rule 8
[15 19] -> existing_rules(8,1);
[1 13] -> existing_rules(8,2);
[5] -> existing_rules(8,3);
[5] -> existing_rules(8,4);
;;; Rule 9
```

```
[15 18] -> existing_rules(9,1);
[1 13] -> existing_rules(9,2);
[4] -> existing_rules(9,3);
[5] -> existing_rules(9,4);
;;; Rule 10
[15 17] -> existing_rules(10,1);
[1 13] -> existing_rules(10,2);
[3] -> existing_rules(10,3);
[5] -> existing_rules(10,4);
;;; Rule 11
[15 16] -> existing_rules(11,1);
[1 13] -> existing_rules(11,2);
[3] -> existing_rules(11,3);
[5] -> existing_rules(11,4);
;;; Rule 12
[14 20] -> existing_rules(12,1);
[1 13] -> existing_rules(12,2);
[5] -> existing_rules(12,3);
[5] -> existing_rules(12,4);
;;; Rule 13
[14 19] -> existing_rules(13,1);
[1 13] -> existing_rules(13,2);
[4] -> existing_rules(13,3);
[5] -> existing_rules(13,4);
;;; Rule 14
[14 18] -> existing_rules(14,1);
[1 13] -> existing_rules(14,2);
[4] -> existing_rules(14,3);
[5] -> existing_rules(14,4);
;;; Rule 15
[14 17] -> existing_rules(15,1);
[1 13] -> existing_rules(15,2);
[3] -> existing_rules(15,3);
[5] -> existing_rules(15,4);
;;; Rule 16
[14 16] -> existing_rules(16,1);
[1 13] -> existing_rules(16,2);
[2] -> existing_rules(16,3);
[5] -> existing_rules(16,4);
;;; Rule 17
[13 20] -> existing_rules(17,1);
[1 13] -> existing_rules(17,2);
[4] -> existing_rules(17,3);
[5] -> existing_rules(17,4);
;;; Rule 18
[13 19] -> existing_rules(18,1);
[1 13] -> existing_rules(18,2);
[4] -> existing_rules(18,3);
[5] -> existing_rules(18,4);
;;; Rule 19
[13 18] -> existing_rules(19,1);
[1 13] -> existing_rules(19,2);
[3] -> existing_rules(19,3);
[5] -> existing_rules(19,4);
;;; Rule 20
[13 17] -> existing_rules(20,1);
[1 13] -> existing_rules(20,2);
[2] -> existing_rules(20,3);
[5] -> existing_rules(20,4);
;;; Rule 21
[13 16] -> existing_rules(21,1);
[1 13] -> existing_rules(21,2);
[2] -> existing_rules(21,3);
[5] -> existing_rules(21,4);
;;; Rule 22
[12 20] -> existing_rules(22,1);
[1 13] -> existing_rules(22,2);
[3] -> existing_rules(22,3);
[5] -> existing_rules(22,4);
;;; Rule 23
[12 19] -> existing_rules(23,1);
[1 13] -> existing_rules(23,2);
[3] -> existing_rules(23,3);
[5] -> existing_rules(23,4);
;;; Rule 24
[12 18] -> existing_rules(24,1);
[1 13] -> existing_rules(24,2);
[2] -> existing_rules(24,3);
[5] -> existing_rules(24,4);
;;; Rule 25
[12 17] -> existing_rules(25,1);
[1 13] -> existing_rules(25,2);
[2] -> existing_rules(25,3);
[5] -> existing_rules(25,4);
;;; Rule 26
[12 16] -> existing_rules(26,1);
[1 13] -> existing_rules(26,2);
[1] -> existing_rules(26,3);
[5] -> existing_rules(26,4);
;;; Rule 27
[11 20] -> existing_rules(27,1);
[1 13] -> existing_rules(27,2);
[3] -> existing_rules(27,3);
[5] -> existing_rules(27,4);
;;; Rule 28
[11 19] -> existing_rules(28,1);
[1 13] -> existing_rules(28,2);
[2] -> existing_rules(28,3);
[5] -> existing_rules(28,4);
;;; Rule 29
[11 18] -> existing_rules(29,1);
[1 13] -> existing_rules(29,2);
[2] -> existing_rules(29,3);
[5] -> existing_rules(29,4);
;;; Rule 30
[11 17] -> existing_rules(30,1);
[1 13] -> existing_rules(30,2);
[1] -> existing_rules(30,3);
[5] -> existing_rules(30,4);
;;; Rule 31
[11 16] -> existing_rules(31,1);
[1 13] -> existing_rules(31,2);
[4] -> existing_rules(31,3);
[6] -> existing_rules(31,4);
;;; Rule 32
[25 30] -> existing_rules(32,1);
[1 13] -> existing_rules(32,2);
[4] -> existing_rules(32,3);
[6] -> existing_rules(32,4);
;;; Rule 33
[25 29] -> existing_rules(33,1);
[1 13] -> existing_rules(33,2);
[4] -> existing_rules(33,3);
[6] -> existing_rules(33,4);
;;; Rule 34
[25 28] -> existing_rules(34,1);
[1 13] -> existing_rules(34,2);
[3] -> existing_rules(34,3);
[6] -> existing_rules(34,4);
;;; Rule 35
[25 27] -> existing_rules(35,1);
[1 13] -> existing_rules(35,2);
[2] -> existing_rules(35,3);
[6] -> existing_rules(35,4);
;;; Rule 36
[25 26] -> existing_rules(36,1);
[1 13] -> existing_rules(36,2);
[2] -> existing_rules(36,3);
[6] -> existing_rules(36,4);
;;; Rule 37
[24 30] -> existing_rules(37,1);
[1 13] -> existing_rules(37,2);
[4] -> existing_rules(37,3);
[6] -> existing_rules(37,4);
;;; Rule 38
[24 29] -> existing_rules(38,1);
[1 13] -> existing_rules(38,2);
[4] -> existing_rules(38,3);
[6] -> existing_rules(38,4);
;;; Rule 39
[24 28] -> existing_rules(39,1);
[1 13] -> existing_rules(39,2);
[3] -> existing_rules(39,3);
[6] -> existing_rules(39,4);
;;; Rule 40
[24 27] -> existing_rules(40,1);
[1 13] -> existing_rules(40,2);
[2] -> existing_rules(40,3);
[6] -> existing_rules(40,4);
;;; Rule 41
[24 26] -> existing_rules(41,1);
[1 13] -> existing_rules(41,2);
[1] -> existing_rules(41,3);
[6] -> existing_rules(41,4);
;;; Rule 42
[23 30] -> existing_rules(42,1);
[1 13] -> existing_rules(42,2);
[3] -> existing_rules(42,3);
```

```
[6] -> existing_rules(42,4);
;;; Rule 43
[23 29] -> existing_rules(43,1);
[1 13] -> existing_rules(43,2);
[3] -> existing_rules(43,3);
[6] -> existing_rules(43,4);
;;; Rule 44
[23 28] -> existing_rules(44,1);
[1 13] -> existing_rules(44,2);
[2] -> existing_rules(44,3);
[6] -> existing_rules(44,4);
;;; Rule 45
[23 27] -> existing_rules(45,1);
[1 13] -> existing_rules(45,2);
[2] -> existing_rules(45,3);
[6] -> existing_rules(45,4);
;;; Rule 46
[23 26] -> existing_rules(46,1);
[1 13] -> existing_rules(46,2);
[1] -> existing_rules(46,3);
[6] -> existing_rules(46,4);
;;; Rule 47
[22 30] -> existing_rules(47,1);
[1 13] -> existing_rules(47,2);
[2] -> existing_rules(47,3);
[6] -> existing_rules(47,4);
;;; Rule 48
[22 29] -> existing_rules(48,1);
[1 13] -> existing_rules(48,2);
[2] -> existing_rules(48,3);
[6] -> existing_rules(48,4);
;;; Rule 49
[22 28] -> existing_rules(49,1);
[1 13] -> existing_rules(49,2);
[2] -> existing_rules(49,3);
[6] -> existing_rules(49,4);
;;; Rule 50
[22 27] -> existing_rules(50,1);
[1 13] -> existing_rules(50,2);
[1] -> existing_rules(50,3);
[6] -> existing_rules(50,4);
;;; Rule 51
[22 26] -> existing_rules(51,1);
[1 13] -> existing_rules(51,2);
[1] -> existing_rules(51,3);
[6] -> existing_rules(51,4);
;;; Rule 52
[21 30] -> existing_rules(52,1);
[1 13] -> existing_rules(52,2);
[2] -> existing_rules(52,3);
[6] -> existing_rules(52,4);
;;; Rule 53
[21 29] -> existing_rules(53,1);
[1 13] -> existing_rules(53,2);
[1] -> existing_rules(53,3);
[6] -> existing_rules(53,4);
;;; Rule 54
[21 28] -> existing_rules(54,1);
[1 13] -> existing_rules(54,2);
[1] -> existing_rules(54,3);
[6] -> existing_rules(54,4);
;;; Rule 55
[21 27] -> existing_rules(55,1);
[1 13] -> existing_rules(55,2);
[1] -> existing_rules(55,3);
[6] -> existing_rules(55,4);
;;; Rule 56
[21 26] -> existing_rules(56,1);
[1 13] -> existing_rules(56,2);
[1] -> existing_rules(56,3);
[6] -> existing_rules(56,4);
;;; Rule 57
[26] -> existing_rules(57,1);
[1] -> existing_rules(57,2);
[4] -> existing_rules(57,3);
[7] -> existing_rules(57,4);
;;; Rule 58
[10] -> existing_rules(58,1);
[1] -> existing_rules(58,2);
[3] -> existing_rules(58,3);
[8] -> existing_rules(58,4);
;;; Rule 59
[9] -> existing_rules(59,1);
[1] -> existing_rules(59,2);
[2] -> existing_rules(59,3);
[8] -> existing_rules(59,4);
;;; Rule 60
[8] -> existing_rules(60,1);
[1] -> existing_rules(60,2);
[2] -> existing_rules(60,3);
[8] -> existing_rules(60,4);
;;; Rule 61
[7] -> existing_rules(61,1);
[1] -> existing_rules(61,2);
[1] -> existing_rules(61,3);
[8] -> existing_rules(61,4);
;;; Rule 62
[6] -> existing_rules(62,1);
[1] -> existing_rules(62,2);
[1] -> existing_rules(62,3);
[8] -> existing_rules(62,4);
;;; Rule 63
[] -> existing_rules(63,1);
[1 5] -> existing_rules(63,2);
[5] -> existing_rules(63,3);
[2] -> existing_rules(63,4);
;;; Rule 64
[] -> existing_rules(64,1);
[1 3] -> existing_rules(64,2);
[2] -> existing_rules(64,3);
[9] -> existing_rules(64,4);
;;; Rule 65
[] -> existing_rules(65,1);
[1 3] -> existing_rules(65,2);
[2] -> existing_rules(65,3);
[10] -> existing_rules(65,4);
;;; Rule 66
[] -> existing_rules(66,1);
[1 3] -> existing_rules(66,2);
[2] -> existing_rules(66,3);
[11] -> existing_rules(66,4);
;;; Rule 67
[] -> existing_rules(67,1);
[1 15] -> existing_rules(67,2);
[2] -> existing_rules(67,3);
[12] -> existing_rules(67,4);
;;; Rule 68
[] -> existing_rules(68,1);
[1 15] -> existing_rules(68,2);
[2] -> existing_rules(68,3);
[13] -> existing_rules(68,4);
;;; Rule 69
[] -> existing_rules(69,1);
[1 5 18 19 21] -> existing_rules(69,2);
[5] -> existing_rules(69,3);
[17] -> existing_rules(69,4);
;;; Rule 70
[] -> existing_rules(70,1);
[1 24] -> existing_rules(70,2);
[5] -> existing_rules(70,3);
[18] -> existing_rules(70,4);
;;; Rule 71
[] -> existing_rules(71,1);
[1 5 25] -> existing_rules(71,2);
[5] -> existing_rules(71,3);
[14] -> existing_rules(71,4);
;;; Rule 72
[35] -> existing_rules(72,1);
[1] -> existing_rules(72,2);
[4] -> existing_rules(72,3);
[6] -> existing_rules(72,4);
;;; Rule 73
[34] -> existing_rules(73,1);
[1] -> existing_rules(73,2);
[4] -> existing_rules(73,3);
[6] -> existing_rules(73,4);
;;; Rule 74
[33] -> existing_rules(74,1);
[1] -> existing_rules(74,2);
[4] -> existing_rules(74,3);
[6] -> existing_rules(74,4);
;;; Rule 75
[32] -> existing_rules(75,1);
[1] -> existing_rules(75,2);
[3] -> existing_rules(75,3);
[6] -> existing_rules(75,4);
;;; Rule 76
[31] -> existing_rules(76,1);
```

```
[1] -> existing_rules(76,2);
[1] -> existing_rules(76,3);
[6] -> existing_rules(76,4);
;;; Rule 77
[31] -> existing_rules(77,1);
[1] -> existing_rules(77,2);
[4] -> existing_rules(77,3);
[9] -> existing_rules(77,4);
;;; Rule 78
[32] -> existing_rules(78,1);
[1] -> existing_rules(78,2);
[3] -> existing_rules(78,3);
[9] -> existing_rules(78,4);
;;; Rule 79
[33] -> existing_rules(79,1);
[1] -> existing_rules(79,2);
[2] -> existing_rules(79,3);
[9] -> existing_rules(79,4);
;;; Rule 80
[34] -> existing_rules(80,1);
[1] -> existing_rules(80,2);
[2] -> existing_rules(80,3);
[9] -> existing_rules(80,4);
;;; Rule 81
[35] -> existing_rules(81,1);
[1] -> existing_rules(81,2);
[1] -> existing_rules(81,3);
[9] -> existing_rules(81,4);
;;; Rule 82
[15] -> existing_rules(82,1);
[1 12] -> existing_rules(82,2);
[5] -> existing_rules(82,3);
[10] -> existing_rules(82,4);
;;; Rule 83
[14] -> existing_rules(83,1);
[1 12] -> existing_rules(83,2);
[4] -> existing_rules(83,3);
[10] -> existing_rules(83,4);
;;; Rule 84
[13] -> existing_rules(84,1);
[1 12] -> existing_rules(84,2);
[4] -> existing_rules(84,3);
[10] -> existing_rules(84,4);
;;; Rule 85
[12] -> existing_rules(85,1);
[1 12] -> existing_rules(85,2);
[3] -> existing_rules(85,3);
[10] -> existing_rules(85,4);
;;; Rule 86
[11] -> existing_rules(86,1);
[1 12] -> existing_rules(86,2);
[2] -> existing_rules(86,3);
[10] -> existing_rules(86,4);
;;; Rule 87
[20] -> existing_rules(87,1);
[1 11] -> existing_rules(87,2);
[5] -> existing_rules(87,3);
[5] -> existing_rules(87,4);
;;; Rule 88
[19] -> existing_rules(88,1);
[1 11] -> existing_rules(88,2);
[4] -> existing_rules(88,3);
[5] -> existing_rules(88,4);
;;; Rule 89
[18] -> existing_rules(89,1);
[1 11] -> existing_rules(89,2);
[3] -> existing_rules(89,3);
[5] -> existing_rules(89,4);
;;; Rule 90
[17] -> existing_rules(90,1);
[1 11] -> existing_rules(90,2);
[2] -> existing_rules(90,3);
[5] -> existing_rules(90,4);
;;; Rule 91
[16] -> existing_rules(91,1);
[1 11] -> existing_rules(91,2);
[1] -> existing_rules(91,3);
[5] -> existing_rules(91,4);
;;; Rule 92
[20] -> existing_rules(92,1);
[1 12] -> existing_rules(92,2);
[5] -> existing_rules(92,3);
[10] -> existing_rules(92,4);
;;; Rule 93
[19] -> existing_rules(93,1);
[1 12] -> existing_rules(93,2);
[4] -> existing_rules(93,3);
[10] -> existing_rules(93,4);
;;; Rule 94
[18] -> existing_rules(94,1);
[1 12] -> existing_rules(94,2);
[3] -> existing_rules(94,3);
[10] -> existing_rules(94,4);
;;; Rule 95
[17] -> existing_rules(95,1);
[1 12] -> existing_rules(95,2);
[2] -> existing_rules(95,3);
[10] -> existing_rules(95,4);
;;; Rule 96
[16] -> existing_rules(96,1);
[1 12] -> existing_rules(96,2);
[1] -> existing_rules(96,3);
[10] -> existing_rules(96,4);
;;; Rule 97
[40] -> existing_rules(97,1);
[] -> existing_rules(97,2);
[3] -> existing_rules(97,3);
[15] -> existing_rules(97,4);
;;; Rule 98
[39] -> existing_rules(98,1);
[] -> existing_rules(98,2);
[2] -> existing_rules(98,3);
[15] -> existing_rules(98,4);
;;; Rule 99
[38] -> existing_rules(99,1);
[] -> existing_rules(99,2);
[2] -> existing_rules(99,3);
[15] -> existing_rules(99,4);
;;; Rule 100
[37] -> existing_rules(100,1);
[] -> existing_rules(100,2);
[1] -> existing_rules(100,3);
[15] -> existing_rules(100,4);
;;; Rule 101
[36] -> existing_rules(101,1);
[] -> existing_rules(101,2);
[1] -> existing_rules(101,3);
[15] -> existing_rules(101,4);
;;; Rule 102
[] -> existing_rules(102,1);
[1 3] -> existing_rules(102,2);
[3] -> existing_rules(102,3);
[11] -> existing_rules(102,4);
;;; Rule 103
[] -> existing_rules(103,1);
[1 27] -> existing_rules(103,2);
[1] -> existing_rules(103,3);
[5] -> existing_rules(103,4);
;;; Rule 104
[] -> existing_rules(104,1);
[1 29] -> existing_rules(104,2);
[1] -> existing_rules(104,3);
[5] -> existing_rules(104,4);
;;; Rule 105
[] -> existing_rules(105,1);
[1 29] -> existing_rules(105,2);
[4] -> existing_rules(105,3);
[10] -> existing_rules(105,4);
;;; Rule 106
[] -> existing_rules(106,1);
[1 31] -> existing_rules(106,2);
[3] -> existing_rules(106,3);
[6] -> existing_rules(106,4);
;;; Rule 107
[] -> existing_rules(107,1);
[1 32] -> existing_rules(107,2);
[3] -> existing_rules(107,3);
[9] -> existing_rules(107,4);
;;; Rule 108
[] -> existing_rules(108,1);
[1 27] -> existing_rules(108,2);
[3] -> existing_rules(108,3);
[15] -> existing_rules(108,4);
;;; Rule 109
[] -> existing_rules(109,1);
[1 20] -> existing_rules(109,2);
[4] -> existing_rules(109,3);
[14] -> existing_rules(109,4);
```

```
;;; Rule 110
[] -> existing_rules(110,1);
[1 9] -> existing_rules(110,2);
[4] -> existing_rules(110,3);
[16] -> existing_rules(110,4);
;;; Rule 111
[5] -> existing_rules(111,1);
[] -> existing_rules(111,2);
[4] -> existing_rules(111,3);
[3] -> existing_rules(111,4);
;;; Rule 112
[4] -> existing_rules(112,1);
[] -> existing_rules(112,2);
[4] -> existing_rules(112,3);
[3] -> existing_rules(112,4);
;;; Rule 113
[3] -> existing_rules(113,1);
[] -> existing_rules(113,2);
[3] -> existing_rules(113,3);
[3] -> existing_rules(113,4);
;;; Rule 114
[2] -> existing_rules(114,1);
[] -> existing_rules(114,2);
[2] -> existing_rules(114,3);
[3] -> existing_rules(114,4);
;;; Rule 115
[1] -> existing_rules(115,1);
[] -> existing_rules(115,2);
[1] -> existing_rules(115,3);
[3] -> existing_rules(115,4);
;;; Rule 116
[31] -> existing_rules(116,1);
[] -> existing_rules(116,2);
[4] -> existing_rules(116,3);
[12] -> existing_rules(116,4);
;;; Rule 117
[32] -> existing_rules(117,1);
[] -> existing_rules(117,2);
[3] -> existing_rules(117,3);
[12] -> existing_rules(117,4);
syscreate("rules2",1,"line") -> filed;
existing_rules -> datafile(filed);
sysclose(filed);

8 -> number_of_fuzzy_premises;
syscreate("noffp2",1,"line") -> filed;
number_of_fuzzy_premises -> datafile(filed);
sysclose(filed);
16 -> number_of_boolean_premises;
syscreate("nofbp2",1,"line") -> filed;
number_of_boolean_premises -> datafile(filed);
sysclose(filed);
18 -> number_of_consequences;
syscreate("noffc2",1,"line") -> filed;
number_of_consequences -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
"substrate_peak_broadening" -> fuzzy_premises(1);
"substrate_asymmetry_in_peak" -> fuzzy_premises(2);
"layer_asymmetry_in_peak" -> fuzzy_premises(3);
"layer_wedge_shaped_peak" -> fuzzy_premises(4);
"interference_fringes" -> fuzzy_premises(5);
"visibility_of_interference_fringes" -> fuzzy_premises(6);
"intensity_of_layer_peak" -> fuzzy_premises(7);
"evidence_of_mismatch" -> fuzzy_premises(8);
syscreate("fp2",1,"line") -> filed;
fuzzy_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
"type_of_structure_is_single_layer" -> boolean_premises(1);
"number_of_peaks_is_one" -> boolean_premises(2);
"number_of_peaks_is_two" -> boolean_premises(3);
"number_of_peaks_is_more_than_two" -> boolean_premises(4);
"number_of_peaks_is_none" -> boolean_premises(5);
"substrate_material_equal_to_layer" -> boolean_premises(6);
"peak_separation_is_low" -> boolean_premises(7);
"layer_split_peak" -> boolean_premises(8);
"layer_integrated_intensity_of_peak_is_zero" -> boolean_premises(9);
"layer_thickness_greater_than_half_micron" -> boolean_premises(10);
"layer_thickness_less_than_5_microns" -> boolean_premises(11);
"peak_splitting_is_zero" -> boolean_premises(12);
"peak_splitting_less_than_three_times_width_of_peak" ->
        boolean_premises(13);
"relaxed_mismatch_is_high" -> boolean_premises(14);
"peak_splitting_is_high" -> boolean_premises(15);
"spacing_of_interference_fringes_is_low" -> boolean_premises(16);
syscreate("bp2",1,"line") -> filed;
boolean_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_consequences]) -> consequences;
"crystal_quality" -> consequences(1);
"characteristic_curve" -> consequences(2);
"misorientation_of_substrate" -> consequences(3);
"bending_of_substrate" -> consequences(4);
"grading_of_the_layer" -> consequences(5);
"layer_is_thick" -> consequences(6);
"change_in_lattice_parameter_with_depth" -> consequences(7);
"layer_is_present_in_the_substrate_peak" -> consequences(8);
"layer_is_thin" -> consequences(9);
"evidence_of_mismatch" -> consequences(10);
"peak_is_outside_the_scan_range" -> consequences(11);
"misoriented_or_mismatched_layer" -> consequences(12);
"multiple_layers" -> consequences(13);
"simulation_or_calibration_chart_is_needed" -> consequences(14);
"relaxation" -> consequences(15);
"incorrect_experiment" -> consequences(16);
"thickness_of_layer" -> consequences(17);
"experimental_mismatch" -> consequences(18);
syscreate("fc2",1,"line") -> filed;
consequences -> datafile(filed);
sysclose(filed);

1 -> angst;
syscreate("angst2",1,"line") -> filed;
angst -> datafile(filed);
sysclose(filed);

number_of_consequences * 5 -> number_of_consequents;
newarray([1 ^number_of_consequents 1 4]) -> potential_rules;
1 -> i;
until i > number_of_consequents
do
  [] -> potential_rules(i,1);
  [] -> potential_rules(i,2);
  [] -> potential_rules(i,3);
  [] -> potential_rules(i,4);
  i + 1 -> i
enduntil;
syscreate("potrule2",1,"line") -> filed;
potential_rules -> datafile(filed);
sysclose(filed);

pr('Record the rule histories and maskings for rules');nl(1);
;;;  Array of lists containing the history of decisions for each rule
newarray([1 ^number_of_rules]) -> rule_history;
;;;  Array used to record if a rule is ASSERTed or UNASSERTed.
newarray([1 ^number_of_rules]) -> masking_of_rule;
1 -> i;
until i > number_of_rules
do
  [] -> rule_history(i);
  "ASSERT" -> masking_of_rule(i);
  i+1 -> i
enduntil;
syscreate("rhist2",1,"line") -> filed;
rule_history -> datafile(filed);
sysclose(filed);
syscreate("rmask2",1,"line") -> filed;
masking_of_rule -> datafile(filed);
sysclose(filed);

148 -> number_of_fuzzy_variables;
syscreate("fuzzno2",1,"line") -> filed;
number_of_fuzzy_variables -> datafile(filed);
sysclose(filed);

pr('Record the membership functions');nl(1);
;;;  MEMBERSHIP FUNCTIONS ARE SEPARATED AND STORED IN
          AN ARRAY OF LISTS
newarray([1 ^number_of_fuzzy_premises]) -> membership_functions;
fuzzy_peak_broadening -> membership_functions(1);
fuzzy_asymmetry -> membership_functions(2);
fuzzy_asymmetry -> membership_functions(3);
fuzzy_wedge_shaped -> membership_functions(4);
fuzzy_interference -> membership_functions(5);
```

```
fuzzy_visibility -> membership_functions(6);
fuzzy_intensity_of_peak -> membership_functions(7);
fuzzy_evidence_of_mismatch -> membership_functions(8);
syscreate("memfunc2",1,"line") -> filed;
membership_functions -> datafile(filed);
sysclose(filed);

pr('Record the variable histories');nl(3);
;;; The history of values is stored for each fuzzy variable
;;; two lists are used to store values that were successful
;;; and values which were unsuccessful
newarray([1 2 1 ^number_of_fuzzy_variables]) -> variable_history;
newarray([1 ^number_of_fuzzy_variables]) -> var_changes;
1 -> i;
until i > number_of_fuzzy_variables
do
  [] -> variable_history(1,i);
  [] -> variable_history(2,i);
  [] -> var_changes(i);
  i+1 -> i
enduntil;
syscreate("varhist2",1,"line") -> filed;
variable_history -> datafile(filed);
sysclose(filed);
syscreate("varch2",1,"line") -> filed;
var_changes -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises 1 2]) -> fuzzy_rules;
[2 3 4 5 6 57 58 59 60 61 62 72 73 74 75 76 77 78 79 80
  81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
  96 97 98 99 100 101 111 112 113 114 115 116 117] -> fuzzy_rules(1,1);
[1 2 3 4 5 106 107 108 109 110 111 137 138 139 140 141 142
  143 144 145 146 112 113 114 115 116 117 118 119 120 121 122 123 124
  125 126 127 128 129 130 131 132 133 134 135 136 147 148] ->
          fuzzy_rules(1,2);
[7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
  31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
  52 53 54 55 56] -> fuzzy_rules(2,1);
[6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
  31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
  54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
  76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
  99 100 101 102 103 104 105] -> fuzzy_rules(2,2);
3 -> i;
until i > number_of_fuzzy_premises
do
  [] -> fuzzy_rules(i,1);
  [] -> fuzzy_rules(i,2);
  i+1 -> i
enduntil;
syscreate("frules2",1,"line") -> filed;
fuzzy_rules -> datafile(filed);
sysclose(filed);

;;; MEMBERSHIP FUNCTION USED BY EACH VARIABLE
newarray([1 ^number_of_fuzzy_variables]) -> mf;
1 -> mf(1);
1 -> mf(2);
1 -> mf(3);
1 -> mf(4);
1 -> mf(5);
2 -> mf(6);
4 -> mf(7);
2 -> mf(8);
4 -> mf(9);
2 -> mf(10);
4 -> mf(11);
2 -> mf(12);
4 -> mf(13);
2 -> mf(14);
4 -> mf(15);
2 -> mf(16);
4 -> mf(17);
2 -> mf(18);
4 -> mf(19);
2 -> mf(20);
4 -> mf(21);
2 -> mf(22);
4 -> mf(23);
2 -> mf(24);
4 -> mf(25);
2 -> mf(26);
4 -> mf(27);
2 -> mf(28);
4 -> mf(29);
2 -> mf(30);
4 -> mf(31);
2 -> mf(32);
4 -> mf(33);
2 -> mf(34);
4 -> mf(35);
2 -> mf(36);
4 -> mf(37);
2 -> mf(38);
4 -> mf(39);
2 -> mf(40);
4 -> mf(41);
2 -> mf(42);
4 -> mf(43);
2 -> mf(44);
4 -> mf(45);
2 -> mf(46);
4 -> mf(47);
2 -> mf(48);
4 -> mf(49);
2 -> mf(50);
4 -> mf(51);
2 -> mf(52);
4 -> mf(53);
2 -> mf(54);
4 -> mf(55);
5 -> mf(56);
6 -> mf(57);
5 -> mf(58);
6 -> mf(59);
5 -> mf(60);
6 -> mf(61);
5 -> mf(62);
6 -> mf(63);
5 -> mf(64);
6 -> mf(65);
5 -> mf(66);
6 -> mf(67);
5 -> mf(68);
6 -> mf(69);
5 -> mf(70);
6 -> mf(71);
5 -> mf(72);
6 -> mf(73);
5 -> mf(74);
6 -> mf(75);
5 -> mf(76);
6 -> mf(77);
5 -> mf(78);
6 -> mf(79);
5 -> mf(80);
6 -> mf(81);
5 -> mf(82);
6 -> mf(83);
5 -> mf(84);
6 -> mf(85);
5 -> mf(86);
6 -> mf(87);
5 -> mf(88);
6 -> mf(89);
5 -> mf(90);
6 -> mf(91);
5 -> mf(92);
6 -> mf(93);
5 -> mf(94);
6 -> mf(95);
5 -> mf(96);
6 -> mf(97);
5 -> mf(98);
6 -> mf(99);
5 -> mf(100);
6 -> mf(101);
5 -> mf(102);
6 -> mf(103);
5 -> mf(104);
6 -> mf(105);
6 -> mf(106);
2 -> mf(107);
2 -> mf(108);
2 -> mf(109);
2 -> mf(110);
2 -> mf(111);
```

```
2 -> mf(112);
2 -> mf(113);
2 -> mf(114);
2 -> mf(115);
2 -> mf(116);
4 -> mf(117);
4 -> mf(118);
4 -> mf(119);
4 -> mf(120);
4 -> mf(121);
4 -> mf(122);
4 -> mf(123);
4 -> mf(124);
4 -> mf(125);
4 -> mf(126);
8 -> mf(127);
8 -> mf(128);
8 -> mf(129);
8 -> mf(130);
8 -> mf(131);
1 -> mf(132);
1 -> mf(133);
1 -> mf(134);
1 -> mf(135);
1 -> mf(136);
7 -> mf(137);
7 -> mf(138);
7 -> mf(139);
7 -> mf(140);
7 -> mf(141);
7 -> mf(142);
7 -> mf(143);
7 -> mf(144);
7 -> mf(145);
7 -> mf(146);
7 -> mf(147);
7 -> mf(148);
syscreate("memf2",1,"line") -> filed;
mf -> datafile(filed);
sysclose(filed);

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1] -> layers;
syscreate("l2",1,"line") -> filed;
layers -> datafile(filed);
sysclose(filed);

;;; Linguistic variables used in fuzzy rules
newarray([1 ^number_of_fuzzy_variables]) -> rule_variable;
'EXTREME' -> rule_variable(1);
'VERY' -> rule_variable(2);
'FAIRLY' -> rule_variable(3);
'SOME' -> rule_variable(4);
'NONE' -> rule_variable(5);
'EXTREME' -> rule_variable(6);
'EXTREME' -> rule_variable(7);
'EXTREME' -> rule_variable(8);
'VERY' -> rule_variable(9);
'EXTREME' -> rule_variable(10);
'FAIRLY' -> rule_variable(11);
'EXTREME' -> rule_variable(12);
'SOME' -> rule_variable(13);
'EXTREME' -> rule_variable(14);
'NONE' -> rule_variable(15);
'VERY' -> rule_variable(16);
'EXTREME' -> rule_variable(17);
'VERY' -> rule_variable(18);
'VERY' -> rule_variable(19);
'VERY' -> rule_variable(20);
'FAIRLY' -> rule_variable(21);
'VERY' -> rule_variable(22);
'SOME' -> rule_variable(23);
'VERY' -> rule_variable(24);
'NONE' -> rule_variable(25);
'FAIRLY' -> rule_variable(26);
'EXTREME' -> rule_variable(27);
'FAIRLY' -> rule_variable(28);
'VERY' -> rule_variable(29);
'FAIRLY' -> rule_variable(30);
'FAIRLY' -> rule_variable(31);
'FAIRLY' -> rule_variable(32);
'SOME' -> rule_variable(33);
'FAIRLY' -> rule_variable(34);
'NONE' -> rule_variable(35);
'SOME' -> rule_variable(36);
'EXTREME' -> rule_variable(37);
'SOME' -> rule_variable(38);
'VERY' -> rule_variable(39);
'SOME' -> rule_variable(40);
'FAIRLY' -> rule_variable(41);
'SOME' -> rule_variable(42);
'SOME' -> rule_variable(43);
'SOME' -> rule_variable(44);
'NONE' -> rule_variable(45);
'NONE' -> rule_variable(46);
'EXTREME' -> rule_variable(47);
'NONE' -> rule_variable(48);
'VERY' -> rule_variable(49);
'NONE' -> rule_variable(50);
'FAIRLY' -> rule_variable(51);
'NONE' -> rule_variable(52);
'SOME' -> rule_variable(53);
'NONE' -> rule_variable(54);
'NONE' -> rule_variable(55);
'EXTREME' -> rule_variable(56);
'EXTREME' -> rule_variable(57);
'EXTREME' -> rule_variable(58);
'VERY' -> rule_variable(59);
'EXTREME' -> rule_variable(60);
'FAIRLY' -> rule_variable(61);
'EXTREME' -> rule_variable(62);
'SOME' -> rule_variable(63);
'EXTREME' -> rule_variable(64);
'NONE' -> rule_variable(65);
'VERY' -> rule_variable(66);
'EXTREME' -> rule_variable(67);
'VERY' -> rule_variable(68);
'VERY' -> rule_variable(69);
'VERY' -> rule_variable(70);
'FAIRLY' -> rule_variable(71);
'VERY' -> rule_variable(72);
'SOME' -> rule_variable(73);
'VERY' -> rule_variable(74);
'NONE' -> rule_variable(75);
'FAIRLY' -> rule_variable(76);
'EXTREME' -> rule_variable(77);
'FAIRLY' -> rule_variable(78);
'VERY' -> rule_variable(79);
'FAIRLY' -> rule_variable(80);
'FAIRLY' -> rule_variable(81);
'FAIRLY' -> rule_variable(82);
'SOME' -> rule_variable(83);
'FAIRLY' -> rule_variable(84);
'NONE' -> rule_variable(85);
'SOME' -> rule_variable(86);
'EXTREME' -> rule_variable(87);
'SOME' -> rule_variable(88);
'VERY' -> rule_variable(89);
'SOME' -> rule_variable(90);
'FAIRLY' -> rule_variable(91);
'SOME' -> rule_variable(92);
'SOME' -> rule_variable(93);
'SOME' -> rule_variable(94);
'NONE' -> rule_variable(95);
'NONE' -> rule_variable(96);
'EXTREME' -> rule_variable(97);
'NONE' -> rule_variable(98);
'VERY' -> rule_variable(99);
'NONE' -> rule_variable(100);
'FAIRLY' -> rule_variable(101);
'NONE' -> rule_variable(102);
'SOME' -> rule_variable(103);
'NONE' -> rule_variable(104);
'NONE' -> rule_variable(105);
'NONE' -> rule_variable(106);
'EXTREME' -> rule_variable(107);
'VERY' -> rule_variable(108);
'FAIRLY' -> rule_variable(109);
'SOME' -> rule_variable(110);
'NONE' -> rule_variable(111);
'EXTREME' -> rule_variable(112);
'VERY' -> rule_variable(113);
'FAIRLY' -> rule_variable(114);
'SOME' -> rule_variable(115);
'NONE' -> rule_variable(116);
'EXTREME' -> rule_variable(117);
```

```
'VERY' -> rule_variable(118);
'FAIRLY' -> rule_variable(119);
'SOME' -> rule_variable(120);
'NONE' -> rule_variable(121);
'EXTREME' -> rule_variable(122);
'VERY' -> rule_variable(123);
'FAIRLY' -> rule_variable(124);
'SOME' -> rule_variable(125);
'NONE' -> rule_variable(126);
'EXTREME' -> rule_variable(127);
'VERY' -> rule_variable(128);
'FAIRLY' -> rule_variable(129);
'SOME' -> rule_variable(130);
'NONE' -> rule_variable(131);
'EXTREME' -> rule_variable(132);
'VERY' -> rule_variable(133);
'FAIRLY' -> rule_variable(134);
'SOME' -> rule_variable(135);
'NONE' -> rule_variable(136);
'EXTREME' -> rule_variable(137);
'VERY' -> rule_variable(138);
'FAIRLY' -> rule_variable(139);
'SOME' -> rule_variable(140);
'NONE' -> rule_variable(141);
'NONE' -> rule_variable(142);
'SOME' -> rule_variable(143);
'FAIRLY' -> rule_variable(144);
'VERY' -> rule_variable(145);
'EXTREME' -> rule_variable(146);
'NONE' -> rule_variable(147);
'SOME' -> rule_variable(148);
syscreate("rvar2",1,"line") -> filed;
rule_variable -> datafile(filed);
sysclose(filed);

;;; ____________________________________
;;;
;;; Multiple Layers Structure
;;; ____________________________________

pr('Multiple Layers Structure');nl(2);
31 -> number_of_rules;
syscreate("rno3",1,"line") -> filed;
number_of_rules -> datafile(filed);
sysclose(filed);
;;; Record the ruleset in a matrix
;;;
;;; Column 1: fuzzy premises
;;; Column 2: boolean premises
;;; Column 3: conclusions
;;; Column 4: Number of conclusion
pr('Record the ruleset');nl(1);
newarray([1 ^number_of_rules 1 4]) -> existing_rules;
;;; Rule 1
[35] -> existing_rules(1,1);
[] -> existing_rules(1,2);
[3] -> existing_rules(1,3);
[4] -> existing_rules(1,4);
;;; Rule 2
[34] -> existing_rules(2,1);
[] -> existing_rules(2,2);
[2] -> existing_rules(2,3);
[4] -> existing_rules(2,4);
;;; Rule 3
[] -> existing_rules(3,1);
[] -> existing_rules(3,2);
[2] -> existing_rules(3,3);
[4] -> existing_rules(3,4);
;;; Rule 4
[] -> existing_rules(4,1);
[] -> existing_rules(4,2);
[1] -> existing_rules(4,3);
[4] -> existing_rules(4,4);
;;; Rule 5
[] -> existing_rules(5,1);
[] -> existing_rules(5,2);
[1] -> existing_rules(5,3);
[4] -> existing_rules(5,4);
;;; Rule 6
[] -> existing_rules(6,1);
[1 3] -> existing_rules(6,2);
[4] -> existing_rules(6,3);
[5] -> existing_rules(6,4);
;;; Rule 7
[] -> existing_rules(7,1);
[1 4] -> existing_rules(7,2);
[4] -> existing_rules(7,3);
[6] -> existing_rules(7,4);
;;; Rule 8
[25] -> existing_rules(8,1);
[1] -> existing_rules(8,2);
[5] -> existing_rules(8,3);
[3] -> existing_rules(8,4);
;;; Rule 9
[24] -> existing_rules(9,1);
[1] -> existing_rules(9,2);
[4] -> existing_rules(9,3);
[3] -> existing_rules(9,4);
;;; Rule 10
[23] -> existing_rules(10,1);
[1] -> existing_rules(10,2);
[3] -> existing_rules(10,3);
[3] -> existing_rules(10,4);
;;; Rule 11
[22] -> existing_rules(11,1);
[1] -> existing_rules(11,2);
[2] -> existing_rules(11,3);
[3] -> existing_rules(11,4);
;;; Rule 12
[21] -> existing_rules(12,1);
[1] -> existing_rules(12,2);
[1] -> existing_rules(12,3);
[3] -> existing_rules(12,4);
;;; Rule 13
[] -> existing_rules(13,1);
[5] -> existing_rules(13,2);
[4] -> existing_rules(13,3);
[7] -> existing_rules(13,4);
;;; Rule 14
[16 30] -> existing_rules(14,1);
[] -> existing_rules(14,2);
[5] -> existing_rules(14,3);
[1] -> existing_rules(14,4);
;;; Rule 15
[17 29] -> existing_rules(15,1);
[] -> existing_rules(15,2);
[4] -> existing_rules(15,3);
[1] -> existing_rules(15,4);
;;; Rule 16
[18 28] -> existing_rules(16,1);
[] -> existing_rules(16,2);
[3] -> existing_rules(16,3);
[1] -> existing_rules(16,4);
;;; Rule 17
[19 27] -> existing_rules(17,1);
[] -> existing_rules(17,2);
[2] -> existing_rules(17,3);
[1] -> existing_rules(17,4);
;;; Rule 18
[20 26] -> existing_rules(18,1);
[] -> existing_rules(18,2);
[1] -> existing_rules(18,3);
[1] -> existing_rules(18,4);
;;; Rule 19
[25] -> existing_rules(19,1);
[1] -> existing_rules(19,2);
[5] -> existing_rules(19,3);
[6] -> existing_rules(19,4);
;;; Rule 20
[24] -> existing_rules(20,1);
[1] -> existing_rules(20,2);
[4] -> existing_rules(20,3);
[6] -> existing_rules(20,4);
;;; Rule 21
[23] -> existing_rules(21,1);
[1] -> existing_rules(21,2);
[3] -> existing_rules(21,3);
[6] -> existing_rules(21,4);
;;; Rule 22
[22] -> existing_rules(22,1);
[1] -> existing_rules(22,2);
[2] -> existing_rules(22,3);
[6] -> existing_rules(22,4);
;;; Rule 23
[21] -> existing_rules(23,1);
[1] -> existing_rules(23,2);
[1] -> existing_rules(23,3);
[6] -> existing_rules(23,4);
```

```
;;; Rule 24
[25] -> existing_rules(24,1);
[4] -> existing_rules(24,2);
[5] -> existing_rules(24,3);
[10] -> existing_rules(24,4);
;;; Rule 25
[24] -> existing_rules(25,1);
[4] -> existing_rules(25,2);
[4] -> existing_rules(25,3);
[10] -> existing_rules(25,4);
;;; Rule 26
[23] -> existing_rules(26,1);
[4] -> existing_rules(26,2);
[3] -> existing_rules(26,3);
[10] -> existing_rules(26,4);
;;; Rule 27
[22] -> existing_rules(27,1);
[4] -> existing_rules(27,2);
[2] -> existing_rules(27,3);
[10] -> existing_rules(27,4);
;;; Rule 28
[21] -> existing_rules(28,1);
[4] -> existing_rules(28,2);
[1] -> existing_rules(28,3);
[10] -> existing_rules(28,4);
;;; Rule 29
[] -> existing_rules(29,1);
[4] -> existing_rules(29,2);
[3] -> existing_rules(29,3);
[8] -> existing_rules(29,4);
;;; Rule 30
[30] -> existing_rules(30,1);
[] -> existing_rules(30,2);
[5] -> existing_rules(30,3);
[2] -> existing_rules(30,4);
;;; Rule 31
[] -> existing_rules(31,1);
[7] -> existing_rules(31,2);
[4] -> existing_rules(31,3);
[9] -> existing_rules(31,4);
syscreate("rules3",1,"line") -> filed;
existing_rules -> datafile(filed);
sysclose(filed);

7 -> number_of_fuzzy_premises;
syscreate("noffp3",1,"line") -> filed;
number_of_fuzzy_premises -> datafile(filed);
sysclose(filed);
4 -> number_of_boolean_premises;
syscreate("nofbp3",1,"line") -> filed;
number_of_boolean_premises -> datafile(filed);
sysclose(filed);
10 -> number_of_consequences;
syscreate("noffc3",1,"line") -> filed;
number_of_consequences -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
"substrate_peak_broadening" -> fuzzy_premises(1);
"substrate_asymmetry_in_peak" -> fuzzy_premises(2);
"multiple_layers_asymmetry_in_peak" -> fuzzy_premises(3);
"multiple_layers_wedge_shaped_peak" -> fuzzy_premises(4);
"interference_fringes" -> fuzzy_premises(5);
"visibility_of_interference_fringes" -> fuzzy_premises(6);
"evidence_of_mismatch" -> fuzzy_premises(7);
syscreate("fp3",1,"line") -> filed;
fuzzy_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
"type_of_structure_is_multiple_layers" -> boolean_premises(1);
"correspondence_between_layers_and_peaks" -> boolean_premises(2);
"number_of_peaks_is_none" -> boolean_premises(3);
"split_substrate_peak" -> boolean_premises(4);
syscreate("bp3",1,"line") -> filed;
boolean_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_consequences]) -> consequences;
"grading" -> consequences(1);
"layer_thickness" -> consequences(2);
"simulation_or_calibration_chart_is_needed" -> consequences(3);
"relaxation" -> consequences(4);
"layers_are_thick" -> consequences(5);
"there_are_hidden_layers_somewhere" -> consequences(6);
"incorrect_experiment" -> consequences(7);
"evidence_of_mismatch" -> consequences(8);
"there_are_thin_layers_at_the_interfaces" -> consequences(9);
"evidence_of_interferometer_structure" -> consequences(10);
syscreate("fc3",1,"line") -> filed;
consequences -> datafile(filed);
sysclose(filed);

1 -> angst;
syscreate("angst3",1,"line") -> filed;
angst -> datafile(filed);
sysclose(filed);

number_of_consequences * 5 -> number_of_consequents;
newarray([1 ^number_of_consequents 1 4]) -> potential_rules;
1 -> i;
until i > number_of_consequents
do
  [] -> potential_rules(i,1);
  [] -> potential_rules(i,2);
  [] -> potential_rules(i,3);
  [] -> potential_rules(i,4);
  i + 1 -> i
enduntil;
syscreate("potrule3",1,"line") -> filed;
potential_rules -> datafile(filed);
sysclose(filed);

pr('Record the rule histories and maskings for rules');nl(1);
;;;  Array of lists containing the history of decisions for each rule
newarray([1 ^number_of_rules]) -> rule_history;
;;;  Array used to record if a rule is ASSERTed or UNASSERTed.
newarray([1 ^number_of_rules]) -> masking_of_rule;
1 -> i;
until i > number_of_rules
do
  [] -> rule_history(i);
  "ASSERT" -> masking_of_rule(i);
  i+1 -> i
enduntil;
syscreate("rhist3",1,"line") -> filed;
rule_history -> datafile(filed);
sysclose(filed);
syscreate("rmask3",1,"line") -> filed;
masking_of_rule -> datafile(filed);
sysclose(filed);

25 -> number_of_fuzzy_variables;
syscreate("fuzzno3",1,"line") -> filed;
number_of_fuzzy_variables -> datafile(filed);
sysclose(filed);

pr('Record the membership functions');nl(1);
;;;  MEMBERSHIP FUNCTIONS ARE SEPARATED AND STORED IN
          AN ARRAY OF LISTS
newarray([1 ^number_of_fuzzy_premises]) -> membership_functions;
fuzzy_peak_broadening -> membership_functions(1);
fuzzy_asymmetry -> membership_functions(2);
fuzzy_asymmetry -> membership_functions(3);
fuzzy_wedge_shaped -> membership_functions(4);
fuzzy_interference -> membership_functions(5);
fuzzy_visibility -> membership_functions(6);
fuzzy_evidence_of_mismatch -> membership_functions(7);
syscreate("memfunc3",1,"line") -> filed;
membership_functions -> datafile(filed);
sysclose(filed);

pr('Record the variable histories');nl(3);
;;;  The history of values is stored for each fuzzy variable
;;;  two lists are used to store values that were successful
;;;  and values which were unsuccessful
newarray([1 2 1 ^number_of_fuzzy_variables]) -> variable_history;
newarray([1 ^number_of_fuzzy_variables]) -> var_changes;
1 -> i;
until i > number_of_fuzzy_variables
do
  [] -> variable_history(1,i);
  [] -> variable_history(2,i);
  [] -> var_changes(i);
  i+1 -> i
enduntil;
syscreate("varhist3",1,"line") -> filed;
variable_history -> datafile(filed);
```

```
sysclose(filed);
syscreate("varch3",1,"line") -> filed;
var_changes -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises 1 2]) -> fuzzy_rules;
[1 2 3 4 5 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28] ->
        fuzzy_rules(1,1);
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25] ->
        fuzzy_rules(1,2);
2 -> i;
until i > number_of_fuzzy_premises
do
  [] -> fuzzy_rules(i,1);
  [] -> fuzzy_rules(i,2);
  i+1 -> i
enduntil;
syscreate("frules3",1,"line") -> filed;
fuzzy_rules -> datafile(filed);
sysclose(filed);

;;;  MEMBERSHIP FUNCTION USED BY EACH VARIABLE
newarray([1 ^number_of_fuzzy_variables]) -> mf;
7 -> mf(1);
7 -> mf(2);
7 -> mf(3);
7 -> mf(4);
7 -> mf(5);
5 -> mf(6);
5 -> mf(7);
5 -> mf(8);
5 -> mf(9);
5 -> mf(10);
1 -> mf(11);
1 -> mf(12);
1 -> mf(13);
1 -> mf(14);
1 -> mf(15);
5 -> mf(16);
5 -> mf(17);
5 -> mf(18);
5 -> mf(19);
5 -> mf(20);
5 -> mf(21);
5 -> mf(22);
5 -> mf(23);
5 -> mf(24);
5 -> mf(25);
syscreate("memf3",1,"line") -> filed;
mf -> datafile(filed);
sysclose(filed);

[2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] -> layers;
syscreate("l3",1,"line") -> filed;
layers -> datafile(filed);
sysclose(filed);

;;;  Linguistic variables used in fuzzy rules
newarray([1 ^number_of_fuzzy_variables]) -> rule_variable;
'EXTREME' -> rule_variable(1);
'VERY' -> rule_variable(2);
'FAIRLY' -> rule_variable(3);
'SOME' -> rule_variable(4);
'NONE' -> rule_variable(5);
'EXTREME' -> rule_variable(6);
'VERY' -> rule_variable(7);
'FAIRLY' -> rule_variable(8);
'SOME' -> rule_variable(9);
'NONE' -> rule_variable(10);
'EXTREME' -> rule_variable(11);
'VERY' -> rule_variable(12);
'FAIRLY' -> rule_variable(13);
'SOME' -> rule_variable(14);
'NONE' -> rule_variable(15);
'EXTREME' -> rule_variable(16);
'VERY' -> rule_variable(17);
'FAIRLY' -> rule_variable(18);
'SOME' -> rule_variable(19);
'NONE' -> rule_variable(20);
'EXTREME' -> rule_variable(21);
'VERY' -> rule_variable(22);
'FAIRLY' -> rule_variable(23);
'SOME' -> rule_variable(24);
'NONE' -> rule_variable(25);

syscreate("rvar3",1,"line") -> filed;
rule_variable -> datafile(filed);
sysclose(filed);

;;;  ---------------------------------------------------
;;;
;;;  MQW Structure and Superlattices
;;;  ---------------------------------------------------

pr('MQW Structure and Superlattices');nl(2);

76 -> number_of_rules;
syscreate("rno4",1,"line") -> filed;
number_of_rules -> datafile(filed);
sysclose(filed);
;;;  Record the ruleset in a matrix
;;;
;;;  Column 1: fuzzy premises
;;;  Column 2: boolean premises
;;;  Column 3: conclusions
;;;  Column 4: Number of conclusion
pr('Record the ruleset');nl(1);
newarray([1 ^number_of_rules 1 4]) -> existing_rules;
;;; Rule 1
[45] -> existing_rules(1,1);
[] -> existing_rules(1,2);
[3] -> existing_rules(1,3);
[5] -> existing_rules(1,4);
;;; Rule 2
[44] -> existing_rules(2,1);
[] -> existing_rules(2,2);
[2] -> existing_rules(2,3);
[5] -> existing_rules(2,4);
;;; Rule 3
[43] -> existing_rules(3,1);
[] -> existing_rules(3,2);
[2] -> existing_rules(3,3);
[5] -> existing_rules(3,4);
;;; Rule 4
[42] -> existing_rules(4,1);
[] -> existing_rules(4,2);
[1] -> existing_rules(4,3);
[5] -> existing_rules(4,4);
;;; Rule 5
[41] -> existing_rules(5,1);
[] -> existing_rules(5,2);
[1] -> existing_rules(5,3);
[5] -> existing_rules(5,4);
;;; Rule 6
[15] -> existing_rules(6,1);
[1] -> existing_rules(6,2);
[5] -> existing_rules(6,3);
[7] -> existing_rules(6,4);
;;; Rule 7
[14] -> existing_rules(7,1);
[1] -> existing_rules(7,2);
[4] -> existing_rules(7,3);
[7] -> existing_rules(7,4);
;;; Rule 8
[13] -> existing_rules(8,1);
[1] -> existing_rules(8,2);
[3] -> existing_rules(8,3);
[7] -> existing_rules(8,4);
;;; Rule 9
[12] -> existing_rules(9,1);
[1] -> existing_rules(9,2);
[2] -> existing_rules(9,3);
[7] -> existing_rules(9,4);
;;; Rule 10
[11] -> existing_rules(10,1);
[1] -> existing_rules(10,2);
[1] -> existing_rules(10,3);
[7] -> existing_rules(10,4);
;;; Rule 11
[15] -> existing_rules(11,1);
[1] -> existing_rules(11,2);
[1] -> existing_rules(11,3);
[6] -> existing_rules(11,4);
;;; Rule 12
[14] -> existing_rules(12,1);
[1] -> existing_rules(12,2);
[1] -> existing_rules(12,3);
[6] -> existing_rules(12,4);
;;; Rule 13
```

```
[13] -> existing_rules(13,1);
[1] -> existing_rules(13,2);
[2] -> existing_rules(13,3);
[6] -> existing_rules(13,4);
;;; Rule 14
[12] -> existing_rules(14,1);
[1] -> existing_rules(14,2);
[3] -> existing_rules(14,3);
[6] -> existing_rules(14,4);
;;; Rule 15
[11] -> existing_rules(15,1);
[1] -> existing_rules(15,2);
[4] -> existing_rules(15,3);
[6] -> existing_rules(15,4);
;;; Rule 16
[] -> existing_rules(16,1);
[1 3] -> existing_rules(16,2);
[5] -> existing_rules(16,3);
[13] -> existing_rules(16,4);
;;; Rule 17
[10] -> existing_rules(17,1);
[1] -> existing_rules(17,2);
[5] -> existing_rules(17,3);
[8] -> existing_rules(17,4);
;;; Rule 18
[9] -> existing_rules(18,1);
[1] -> existing_rules(18,2);
[4] -> existing_rules(18,3);
[8] -> existing_rules(18,4);
;;; Rule 19
[8] -> existing_rules(19,1);
[1] -> existing_rules(19,2);
[3] -> existing_rules(19,3);
[8] -> existing_rules(19,4);
;;; Rule 20
[7] -> existing_rules(20,1);
[1] -> existing_rules(20,2);
[2] -> existing_rules(20,3);
[8] -> existing_rules(20,4);
;;; Rule 21
[6] -> existing_rules(21,1);
[1] -> existing_rules(21,2);
[1] -> existing_rules(21,3);
[8] -> existing_rules(21,4);
;;; Rule 22
[1] -> existing_rules(22,1);
[1] -> existing_rules(22,2);
[5] -> existing_rules(22,3);
[9] -> existing_rules(22,4);
;;; Rule 23
[2] -> existing_rules(23,1);
[1] -> existing_rules(23,2);
[4] -> existing_rules(23,3);
[9] -> existing_rules(23,4);
;;; Rule 24
[3] -> existing_rules(24,1);
[1] -> existing_rules(24,2);
[3] -> existing_rules(24,3);
[9] -> existing_rules(24,4);
;;; Rule 25
[4] -> existing_rules(25,1);
[1] -> existing_rules(25,2);
[2] -> existing_rules(25,3);
[9] -> existing_rules(25,4);
;;; Rule 26
[5] -> existing_rules(26,1);
[1] -> existing_rules(26,2);
[1] -> existing_rules(26,3);
[9] -> existing_rules(26,4);
;;; Rule 27
[] -> existing_rules(27,1);
[1 5] -> existing_rules(27,2);
[4] -> existing_rules(27,3);
[10] -> existing_rules(27,4);
;;; Rule 28
[] -> existing_rules(28,1);
[1 7] -> existing_rules(28,2);
[4] -> existing_rules(28,3);
[8] -> existing_rules(28,4);
;;; Rule 29
[10 15] -> existing_rules(29,1);
[1] -> existing_rules(29,2);
[1] -> existing_rules(29,3);
[2] -> existing_rules(29,4);
;;; Rule 30
[9 15] -> existing_rules(30,1);
[1] -> existing_rules(30,2);
[1] -> existing_rules(30,3);
[2] -> existing_rules(30,4);
;;; Rule 31
[8 15] -> existing_rules(31,1);
[1] -> existing_rules(31,2);
[3] -> existing_rules(31,3);
[2] -> existing_rules(31,4);
;;; Rule 32
[7 15] -> existing_rules(32,1);
[1] -> existing_rules(32,2);
[5] -> existing_rules(32,3);
[2] -> existing_rules(32,4);
;;; Rule 33
[6 15] -> existing_rules(33,1);
[1] -> existing_rules(33,2);
[5] -> existing_rules(33,3);
[2] -> existing_rules(33,4);
;;; Rule 34
[10 14] -> existing_rules(34,1);
[1] -> existing_rules(34,2);
[1] -> existing_rules(34,3);
[2] -> existing_rules(34,4);
;;; Rule 35
[9 14] -> existing_rules(35,1);
[1] -> existing_rules(35,2);
[1] -> existing_rules(35,3);
[2] -> existing_rules(35,4);
;;; Rule 36
[8 14] -> existing_rules(36,1);
[1] -> existing_rules(36,2);
[3] -> existing_rules(36,3);
[2] -> existing_rules(36,4);
;;; Rule 37
[7 14] -> existing_rules(37,1);
[1] -> existing_rules(37,2);
[5] -> existing_rules(37,3);
[2] -> existing_rules(37,4);
;;; Rule 38
[6 14] -> existing_rules(38,1);
[1] -> existing_rules(38,2);
[5] -> existing_rules(38,3);
[2] -> existing_rules(38,4);
;;; Rule 39
[10 13] -> existing_rules(39,1);
[1] -> existing_rules(39,2);
[1] -> existing_rules(39,3);
[2] -> existing_rules(39,4);
;;; Rule 40
[9 13] -> existing_rules(40,1);
[1] -> existing_rules(40,2);
[1] -> existing_rules(40,3);
[2] -> existing_rules(40,4);
;;; Rule 41
[8 13] -> existing_rules(41,1);
[1] -> existing_rules(41,2);
[3] -> existing_rules(41,3);
[2] -> existing_rules(41,4);
;;; Rule 42
[7 13] -> existing_rules(42,1);
[1] -> existing_rules(42,2);
[3] -> existing_rules(42,3);
[2] -> existing_rules(42,4);
;;; Rule 43
[6 13] -> existing_rules(43,1);
[1] -> existing_rules(43,2);
[3] -> existing_rules(43,3);
[2] -> existing_rules(43,4);
;;; Rule 44
[10 12] -> existing_rules(44,1);
[1] -> existing_rules(44,2);
[1] -> existing_rules(44,3);
[2] -> existing_rules(44,4);
;;; Rule 45
[9 12] -> existing_rules(45,1);
[1] -> existing_rules(45,2);
[1] -> existing_rules(45,3);
[2] -> existing_rules(45,4);
;;; Rule 46
[8 12] -> existing_rules(46,1);
[1] -> existing_rules(46,2);
[1] -> existing_rules(46,3);
```

```
[2] -> existing_rules(46,4);
;;; Rule 47
[7 12] -> existing_rules(47,1);
[1] -> existing_rules(47,2);
[2] -> existing_rules(47,3);
[2] -> existing_rules(47,4);
;;; Rule 48
[6 12] -> existing_rules(48,1);
[1] -> existing_rules(48,2);
[2] -> existing_rules(48,3);
[2] -> existing_rules(48,4);
;;; Rule 49
[10 11] -> existing_rules(49,1);
[1] -> existing_rules(49,2);
[1] -> existing_rules(49,3);
[2] -> existing_rules(49,4);
;;; Rule 50
[9 11] -> existing_rules(50,1);
[1] -> existing_rules(50,2);
[1] -> existing_rules(50,3);
[2] -> existing_rules(50,4);
;;; Rule 51
[8 11] -> existing_rules(51,1);
[1] -> existing_rules(51,2);
[1] -> existing_rules(51,3);
[2] -> existing_rules(51,4);
;;; Rule 52
[7 11] -> existing_rules(52,1);
[1] -> existing_rules(52,2);
[1] -> existing_rules(52,3);
[2] -> existing_rules(52,4);
;;; Rule 53
[6 11] -> existing_rules(53,1);
[1] -> existing_rules(53,2);
[1] -> existing_rules(53,3);
[2] -> existing_rules(53,4);
;;; Rule 54
[1] -> existing_rules(54,1);
[1] -> existing_rules(54,2);
[5] -> existing_rules(54,3);
[11] -> existing_rules(54,4);
;;; Rule 55
[2] -> existing_rules(55,1);
[1] -> existing_rules(55,2);
[3] -> existing_rules(55,3);
[11] -> existing_rules(55,4);
;;; Rule 56
[3] -> existing_rules(56,1);
[1] -> existing_rules(56,2);
[2] -> existing_rules(56,3);
[11] -> existing_rules(56,4);
;;; Rule 57
[4] -> existing_rules(57,1);
[1] -> existing_rules(57,2);
[1] -> existing_rules(57,3);
[11] -> existing_rules(57,4);
;;; Rule 58
[5] -> existing_rules(58,1);
[1] -> existing_rules(58,2);
[1] -> existing_rules(58,3);
[11] -> existing_rules(58,4);
;;; Rule 59
[] -> existing_rules(59,1);
[1 10] -> existing_rules(59,2);
[1] -> existing_rules(59,3);
[1] -> existing_rules(59,4);
;;; Rule 60
[] -> existing_rules(60,1);
[1 11] -> existing_rules(60,2);
[5] -> existing_rules(60,3);
[13] -> existing_rules(60,4);
;;; Rule 61
[] -> existing_rules(61,1);
[1 13] -> existing_rules(61,2);
[5] -> existing_rules(61,3);
[4] -> existing_rules(61,4);
;;; Rule 62
[] -> existing_rules(62,1);
[1 11] -> existing_rules(62,2);
[1] -> existing_rules(62,3);
[1] -> existing_rules(62,4);
;;; Rule 63
[] -> existing_rules(63,1);
[1 9] -> existing_rules(63,2);
[5] -> existing_rules(63,3);
[14] -> existing_rules(63,4);
;;; Rule 64
[] -> existing_rules(64,1);
[1 11] -> existing_rules(64,2);
[3] -> existing_rules(64,3);
[1] -> existing_rules(64,4);
;;; Rule 65
[35] -> existing_rules(65,1);
[1 15] -> existing_rules(65,2);
[5] -> existing_rules(65,3);
[7] -> existing_rules(65,4);
;;; Rule 66
[34] -> existing_rules(66,1);
[1 15] -> existing_rules(66,2);
[4] -> existing_rules(66,3);
[7] -> existing_rules(66,4);
;;; Rule 67
[33] -> existing_rules(67,1);
[1 15] -> existing_rules(67,2);
[3] -> existing_rules(67,3);
[7] -> existing_rules(67,4);
;;; Rule 68
[32] -> existing_rules(68,1);
[1 15] -> existing_rules(68,2);
[2] -> existing_rules(68,3);
[7] -> existing_rules(68,4);
;;; Rule 69
[31] -> existing_rules(69,1);
[1 15] -> existing_rules(69,2);
[1] -> existing_rules(69,3);
[7] -> existing_rules(69,4);
;;; Rule 70
[] -> existing_rules(70,1);
[1 7 15] -> existing_rules(70,2);
[4] -> existing_rules(70,3);
[8] -> existing_rules(70,4);
;;; Rule 71
[] -> existing_rules(71,1);
[17] -> existing_rules(71,2);
[4] -> existing_rules(71,3);
[12] -> existing_rules(71,4);
;;; Rule 72
[20] -> existing_rules(72,1);
[] -> existing_rules(72,2);
[4] -> existing_rules(72,3);
[3] -> existing_rules(72,4);
;;; Rule 73
[19] -> existing_rules(73,1);
[] -> existing_rules(73,2);
[4] -> existing_rules(73,3);
[3] -> existing_rules(73,4);
;;; Rule 74
[18] -> existing_rules(74,1);
[] -> existing_rules(74,2);
[3] -> existing_rules(74,3);
[3] -> existing_rules(74,4);
;;; Rule 75
[17] -> existing_rules(75,1);
[] -> existing_rules(75,2);
[2] -> existing_rules(75,3);
[3] -> existing_rules(75,4);
;;; Rule 76
[16] -> existing_rules(76,1);
[] -> existing_rules(76,2);
[1] -> existing_rules(76,3);
[3] -> existing_rules(76,4);
syscreate("rules4",1,"line") -> filed;
existing_rules -> datafile(filed);
sysclose(filed);

9 -> number_of_fuzzy_premises;
syscreate("noffp4",1,"line") ->filed;
number_of_fuzzy_premises -> datafile(filed);
sysclose(filed);
9 -> number_of_boolean_premises;
syscreate("nofbp4",1,"line") ->filed;
number_of_boolean_premises -> datafile(filed);
sysclose(filed);
14 -> number_of_consequences;
syscreate("noffc4",1,"line") ->filed;
number_of_consequences -> datafile(filed);
sysclose(filed);
```

```
newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
"satellite_visibility" -> fuzzy_premises(1);
"satellite_asymmetry_of_plus_and_minus_peaks" -> fuzzy_premises(2);
"separation_of_satellite_peaks" -> fuzzy_premises(3);
"substrate_peak_broadening" -> fuzzy_premises(4);
"substrate_asymmetry_in_peak" -> fuzzy_premises(5);
"zero_order_asymmetry_in_peak" -> fuzzy_premises(6);
"interference_fringes" -> fuzzy_premises(7);
"visibility_of_interference_fringes" -> fuzzy_premises(8);
"evidence_of_mismatch" -> fuzzy_premises(9);
syscreate("fp4",1,"line") -> filed;
fuzzy_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
"type_of_structure_is_MQW" -> boolean_premises(1);
"satellite_spacing_greater_than_zero" -> boolean_premises(2);
"satellite_subsidiary_interference_effects" -> boolean_premises(3);
"satellite_broadening_of_higher_order_peaks" -> boolean_premises(4);
 "satellite_relative_widths_of_peaks_greater_than_zero" ->
          boolean_premises(5);
 "satellite_relative_integrated_intensities_greater_than_zero" ->
          boolean_premises(6);
 "thickness_of_superlattice_greater_than_half_micron"  ->
          boolean_premises(7);
"type_of_structure_has_additional_layers" -> boolean_premises(8);
"number_of_peaks_is_none" -> boolean_premises(9);
syscreate("bp4",1,"line") -> filed;
boolean_premises -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_consequences]) -> consequences;
"crystal_quality" -> consequences(1);
"characteristic_curve" -> consequences(2);
"misorientation_of_substrate" -> consequences(3);
"simulation_or_calibration_chart_is_needed" -> consequences(4);
"relaxation" -> consequences(5);
"layers_are_thick" -> consequences(6);
"layers_are_thin" -> consequences(7);
"grading_or_dispersion_of_layer_thicknesses" -> consequences(8);
"grading_occurs_through_AB_layers" -> consequences(9);
"large_overall_layer_thickness" -> consequences(10);
"layers_are_not_uniform" -> consequences(11);
"incorrect_experiment" -> consequences(12);
"period_of_superlattice" -> consequences(13);
"period_dispersion" -> consequences(14);
syscreate("fc4",1,"line") -> filed;
consequences -> datafile(filed);
sysclose(filed);

1 -> angst;
syscreate("angst4",1,"line") -> filed;
angst -> datafile(filed);
sysclose(filed);

number_of_consequences * 5 -> number_of_consequents;
newarray([1 ^number_of_consequents 1 4]) -> potential_rules;
1 -> i;
until i > number_of_consequents
do
  [] -> potential_rules(i,1);
  [] -> potential_rules(i,2);
  [] -> potential_rules(i,3);
  [] -> potential_rules(i,4);
  i + 1 -> i
enduntil;
syscreate("potrule4",1,"line") -> filed;
potential_rules -> datafile(filed);
sysclose(filed);

pr('Record the rule histories and maskings for rules');nl(1);
;;; Array of lists containing the history of decisions for each rule
newarray([1 ^number_of_rules]) -> rule_history;
;;; Array used to record if a rule is ASSERTed or UNASSERTed.
newarray([1 ^number_of_rules]) -> masking_of_rule;
1 -> i;
until i > number_of_rules
do
  [] -> rule_history(i);
  "ASSERT" -> masking_of_rule(i);
  i+1 -> i
enduntil;
syscreate("rhist4",1,"line") -> filed;
rule_history -> datafile(filed);
sysclose(filed);
syscreate("rmask4",1,"line") -> filed;
masking_of_rule -> datafile(filed);
sysclose(filed);

90 -> number_of_fuzzy_variables;
syscreate("fuzzno4",1,"line") -> filed;
number_of_fuzzy_variables -> datafile(filed);
sysclose(filed);

pr('Record the membership functions');nl(1);
;;; MEMBERSHIP FUNCTIONS ARE SEPARATED AND STORED IN
          AN ARRAY OF LISTS
newarray([1 ^number_of_fuzzy_premises]) -> membership_functions;
fuzzy_visibility -> membership_functions(1);
fuzzy_plus_minus_asymmetry -> membership_functions(2);
fuzzy_spacing_of_peaks -> membership_functions(3);
fuzzy_peak_broadening -> membership_functions(4);
fuzzy_asymmetry -> membership_functions(5);
fuzzy_asymmetry -> membership_functions(6);
fuzzy_interference -> membership_functions(7);
fuzzy_visibility -> membership_functions(8);
fuzzy_evidence_of_mismatch -> membership_functions(9);
syscreate("memfunc4",1,"line") -> filed;
membership_functions -> datafile(filed);
sysclose(filed);

pr('Record the variable histories');nl(3);
;;; The history of values is stored for each fuzzy variable
;;; two lists are used to store values that were successful
;;; and values which were unsuccessful
newarray([1 2 1 ^number_of_fuzzy_variables]) -> variable_history;
newarray([1 ^number_of_fuzzy_variables]) -> var_changes;
1 -> i;
until i > number_of_fuzzy_variables
do
  [] -> variable_history(1,i);
  [] -> variable_history(2,i);
  [] -> var_changes(i);
  i+1 -> i
enduntil;
syscreate("varhist4",1,"line") -> filed;
variable_history -> datafile(filed);
sysclose(filed);
syscreate("varch4",1,"line") -> filed;
var_changes -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_fuzzy_premises 1 2]) -> fuzzy_rules;
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 18 19 20 21 22 23 24 25
  26 54 55 56 57 58 65 66 67 68 69 72 73 74 75 76] -> fuzzy_rules(1,1);
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
  76 77 78 79 80 81 82 83 84 85 86 87 88 89 90] -> fuzzy_rules(1,2);
[29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
  51 52 53] -> fuzzy_rules(2,1);
[26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
  49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
  72 73 74 75 76] -> fuzzy_rules(2,2);
3 -> i;
until i > number_of_fuzzy_premises
do
  [] -> fuzzy_rules(i,1);
  [] -> fuzzy_rules(i,2);
  i+1 -> i
enduntil;
syscreate("frules4",1,"line") -> filed;
fuzzy_rules -> datafile(filed);
sysclose(filed);

;;; MEMBERSHIP FUNCTION USED BY EACH VARIABLE
newarray([1 ^number_of_fuzzy_variables]) -> mf;
9 -> mf(1);
9 -> mf(2);
9 -> mf(3);
9 -> mf(4);
9 -> mf(5);
3 -> mf(6);
3 -> mf(7);
3 -> mf(8);
3 -> mf(9);
3 -> mf(10);
3 -> mf(11);
3 -> mf(12);
3 -> mf(13);
```

```
3 -> mf(14);
3 -> mf(15);
4 -> mf(16);
4 -> mf(17);
4 -> mf(18);
4 -> mf(19);
4 -> mf(20);
1 -> mf(21);
1 -> mf(22);
1 -> mf(23);
1 -> mf(24);
1 -> mf(25);
3 -> mf(26);
2 -> mf(27);
3 -> mf(28);
2 -> mf(29);
3 -> mf(30);
2 -> mf(31);
3 -> mf(32);
2 -> mf(33);
3 -> mf(34);
2 -> mf(35);
3 -> mf(36);
2 -> mf(37);
3 -> mf(38);
2 -> mf(39);
3 -> mf(40);
2 -> mf(41);
3 -> mf(42);
2 -> mf(43);
3 -> mf(44);
2 -> mf(45);
3 -> mf(46);
2 -> mf(47);
3 -> mf(48);
2 -> mf(49);
3 -> mf(50);
2 -> mf(51);
3 -> mf(52);
2 -> mf(53);
3 -> mf(54);
2 -> mf(55);
3 -> mf(56);
2 -> mf(57);
3 -> mf(58);
2 -> mf(59);
3 -> mf(60);
2 -> mf(61);
3 -> mf(62);
2 -> mf(63);
3 -> mf(64);
2 -> mf(65);
3 -> mf(66);
2 -> mf(67);
3 -> mf(68);
2 -> mf(69);
3 -> mf(70);
2 -> mf(71);
3 -> mf(72);
2 -> mf(73);
3 -> mf(74);
2 -> mf(75);
1 -> mf(76);
1 -> mf(77);
1 -> mf(78);
1 -> mf(79);
1 -> mf(80);
7 -> mf(81);
7 -> mf(82);
7 -> mf(83);
7 -> mf(84);
7 -> mf(85);
4 -> mf(86);
4 -> mf(87);
4 -> mf(88);
4 -> mf(89);
4 -> mf(90);
syscreate("meanf4",1,"line") -> filed;
mf -> datafile(filed);
sysclose(filed);

[2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] -> layers;
```

```
syscreate("l4",1,"line") -> filed;
layers -> datafile(filed);
sysclose(filed);

;;; Linguistic variables used in fuzzy rules
newarray([1 ^number_of_fuzzy_variables]) -> rule_variable;
'EXTREME' -> rule_variable(1);
'VERY' -> rule_variable(2);
'FAIRLY' -> rule_variable(3);
'SOME' -> rule_variable(4);
'NONE' -> rule_variable(5);
'EXTREME' -> rule_variable(6);
'VERY' -> rule_variable(7);
'FAIRLY' -> rule_variable(8);
'SOME' -> rule_variable(9);
'NONE' -> rule_variable(10);
'EXTREME' -> rule_variable(11);
'VERY' -> rule_variable(12);
'FAIRLY' -> rule_variable(13);
'SOME' -> rule_variable(14);
'NONE' -> rule_variable(15);
'EXTREME' -> rule_variable(16);
'VERY' -> rule_variable(17);
'FAIRLY' -> rule_variable(18);
'SOME' -> rule_variable(19);
'NONE' -> rule_variable(20);
'EXTREME' -> rule_variable(21);
'VERY' -> rule_variable(22);
'FAIRLY' -> rule_variable(23);
'SOME' -> rule_variable(24);
'NONE' -> rule_variable(25);
'EXTREME' -> rule_variable(26);
'EXTREME' -> rule_variable(27);
'EXTREME' -> rule_variable(28);
'VERY' -> rule_variable(29);
'EXTREME' -> rule_variable(30);
'FAIRLY' -> rule_variable(31);
'EXTREME' -> rule_variable(32);
'SOME' -> rule_variable(33);
'EXTREME' -> rule_variable(34);
'NONE' -> rule_variable(35);
'VERY' -> rule_variable(36);
'EXTREME' -> rule_variable(37);
'VERY' -> rule_variable(38);
'VERY' -> rule_variable(39);
'VERY' -> rule_variable(40);
'FAIRLY' -> rule_variable(41);
'VERY' -> rule_variable(42);
'SOME' -> rule_variable(43);
'VERY' -> rule_variable(44);
'NONE' -> rule_variable(45);
'FAIRLY' -> rule_variable(46);
'EXTREME' -> rule_variable(47);
'FAIRLY' -> rule_variable(48);
'VERY' -> rule_variable(49);
'FAIRLY' -> rule_variable(50);
'FAIRLY' -> rule_variable(51);
'FAIRLY' -> rule_variable(52);
'SOME' -> rule_variable(53);
'FAIRLY' -> rule_variable(54);
'NONE' -> rule_variable(55);
'SOME' -> rule_variable(56);
'EXTREME' -> rule_variable(57);
'SOME' -> rule_variable(58);
'VERY' -> rule_variable(59);
'SOME' -> rule_variable(60);
'FAIRLY' -> rule_variable(61);
'SOME' -> rule_variable(62);
'SOME' -> rule_variable(63);
'SOME' -> rule_variable(64);
'NONE' -> rule_variable(65);
'NONE' -> rule_variable(66);
'EXTREME' -> rule_variable(67);
'NONE' -> rule_variable(68);
'VERY' -> rule_variable(69);
'NONE' -> rule_variable(70);
'FAIRLY' -> rule_variable(71);
'NONE' -> rule_variable(72);
'SOME' -> rule_variable(73);
'NONE' -> rule_variable(74);
'NONE' -> rule_variable(75);
'EXTREME' -> rule_variable(76);
'VERY' -> rule_variable(77);
'FAIRLY' -> rule_variable(78);
```

```
'SOME' -> rule_variable(79);
'NONE' -> rule_variable(80);
'EXTREME' -> rule_variable(81);
'VERY' -> rule_variable(82);
'FAIRLY' -> rule_variable(83);
'SOME' -> rule_variable(84);
'NONE' -> rule_variable(85);
'EXTREME' -> rule_variable(86);
'VERY' -> rule_variable(87);
'FAIRLY' -> rule_variable(88);
'SOME' -> rule_variable(89);
'NONE' -> rule_variable(90);
syscreate("rvar4",1,"line") -> filed;
rule_variable -> datafile(filed);
sysclose(filed);

;;; ---------- ----------
;;;
;;; VARIABLES DESCRIBING THE EXPERTS
;;; ---------- ----------

pr('VARIABLES DESCRIBING THE EXPERTS');nl(2);
4 -> number_of_experts;
syscreate("noexp",1,"line") -> filed;
number_of_experts -> datafile(filed);
sysclose(filed);

pr('Constructing connection matrices for DKB');nl(1);
make_instance([CONNECTION_MATRIX]) -> connection_matrix;

;;; DKB Connection Matrices

pr(' substrate only connection matrix');nl(1);
;;; Substrate Only
19 -> number_of_rules;
[19] -> rules_for_substrate_only;
syscreate("lofr1s1",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_substrate_only);
connection_matrix <- save_to_file(1,1);

pr(' single layer connection matrix');nl(1);
;;; Single Layer
117 -> number_of_rules;
[7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 67 68 69 70 71 109 110 116 117] ->
        rules_for_single_layer;
syscreate("lofr1s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_single_layer);
connection_matrix <- save_to_file(1,2);

pr(' multiple layers connection matrix');nl(1);
;;; Multiple Layers
31 -> number_of_rules;
[13] -> rules_for_multiple_layers;
syscreate("lofr1s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_multiple_layers);
connection_matrix <- save_to_file(1,3);

pr(' MQW and superlattice connection matrix');nl(2);
;;; MQW Structure and Superlattices
76 -> number_of_rules;
[6 7 8 9 10 11 12 13 14 15 16 22 23 24 25 26 28 59 60 61 62 63 71] ->
        rules_for_MQW;
syscreate("lofr1s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_MQW);
connection_matrix <- save_to_file(1,4);

pr('Constructing connection matrices for BKT');nl(1);

;;; BKT Connection Matrices

pr(' substrate only connection matrix');nl(1);
;;; Substrate Only
19 -> number_of_rules;
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19] ->
        rules_for_substrate_only;
syscreate("lofr2s1",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_substrate_only);
connection_matrix <- save_to_file(2,1);

pr(' single layer connection matrix');nl(1);
;;; Single Layer
117 -> number_of_rules;
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 82 83 84 85 86 87 88
 89 90 100 101 102 103 104 106 107 108 110 111 112 113 114 115] ->
        rules_for_single_layer;
syscreate("lofr2s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_single_layer);
connection_matrix <- save_to_file(2,2);

pr(' multiple layers connection matrix');nl(1);
;;; Multiple Layers
31 -> number_of_rules;
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30 31] -> rules_for_multiple_layers;
syscreate("lofr2s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_multiple_layers);
connection_matrix <- save_to_file(2,3);

pr(' MQW and superlattice connection matrix');nl(2);
;;; MQW Structure and Superlattices
76 -> number_of_rules;
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 57 58 64 65 66 67 68 69 70 71 72 73 74 75 76] ->
        rules_for_MQW;
syscreate("lofr2s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_MQW);
connection_matrix <- save_to_file(2,4);

pr('Constructing connection matrices for NL');nl(1);


;;; NL Connection Matrices

pr(' substrate only connection matrix');nl(1);
;;; Substrate Only
19 -> number_of_rules;
[1 2 3 4 5 19] -> rules_for_substrate_only;
syscreate("lofr3s1",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_substrate_only);
connection_matrix <- save_to_file(3,1);

pr(' single layer connection matrix');nl(1);
;;; Single Layer
117 -> number_of_rules;
[72 73 74 75 76 77 78 79 80 81 116 117] -> rules_for_single_layer;
syscreate("lofr3s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix <- create_a_connection_matrix(number_of_rules,
        rules_for_single_layer);
connection_matrix <- save_to_file(3,2);

pr(' multiple layers connection matrix');nl(1);
```

```
;;; Multiple Layers
31 -> number_of_rules;
[13] -> rules_for_multiple_layers;
syscreate("lofr3s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_multiple_layers);
connection_matrix <- save_to_file(3,3);

pr('  MQW and superlattice connection matrix');nl(3);
;;; MQW Structure and Superlattices
76 -> number_of_rules;
[54 55 56 57 58 59 71] -> rules_for_MQW;
syscreate("lofr3s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_MQW);
connection_matrix <- save_to_file(3,4);

;;; CRT Connection Matrices

pr('  substrate only connection matrix');nl(1);
;;; Substrate Only
19 -> number_of_rules;
[19] -> rules_for_substrate_only;
syscreate("lofr4s1",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_substrate_only);
connection_matrix <- save_to_file(4,1);

pr('  single layer connection matrix');nl(1);
;;; Single Layer
117 -> number_of_rules;
[1 63 70 97 98 99 100 101 110] -> rules_for_single_layer;
syscreate("lofr4s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_single_layer);
connection_matrix <- save_to_file(4,2);

pr('  multiple layers connection matrix');nl(1);
;;; Multiple Layers
31 -> number_of_rules;
[13] -> rules_for_multiple_layers;
syscreate("lofr4s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_multiple_layers);
connection_matrix <- save_to_file(4,3);

pr('  MQW and superlattice connection matrix');nl(3);
;;; MQW Structure and Superlattices
76 -> number_of_rules;
[] -> rules_for_MQW;
syscreate("lofr4s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_MQW);
connection_matrix <- save_to_file(4,4);

pr('Constructing connection matrices for dummy expert DE');nl(1);

;;; DE Connection Matrices

pr('  substrate only connection matrix');nl(1);
;;; Substrate Only Structure
19 -> number_of_rules;
[] -> rules_for_substrate_only;
syscreate("lofr5s1",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_substrate_only);
connection_matrix <- save_to_file(5,1);

pr('  single layer connection matrix');nl(1);
;;; Single Layer Structure
117 -> number_of_rules;
[] -> rules_for_single_layer;
syscreate("lofr5s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_single_layer);
connection_matrix <- save_to_file(5,2);

pr('  multiple layers connection matrix');nl(1);
;;; Multiple Layers Structure
31 -> number_of_rules;
[] -> rules_for_multiple_layers;
syscreate("lofr5s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_multiple_layers);
connection_matrix <- save_to_file(5,3);

pr('  MQW and superlattice connection matrix');nl(3);
;;; MQW Structure and Superlattices
76 -> number_of_rules;
[] -> rules_for_MQW;
syscreate("lofr5s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix    <-    create_a_connection_matrix(number_of_rules,
          rules_for_MQW);
connection_matrix <- save_to_file(5,4);

cancel connection_matrix;

;;; ------------------------------------------------
;;;
;;; GENERAL CONTROL VARIABLES FOR CONNECTION MATRICES
;;; ------------------------------------------------

pr('GENERAL CONTROL VARIABLES FOR CONNECTION
          MATRICES');nl(2);
pr('credibility weights');nl(1);
newarray([1 ^number_of_experts]) -> cred_weight;
0.7 -> cred_weight(1);
0.7 -> cred_weight(2);
0.6 -> cred_weight(3);
0.6 -> cred_weight(4);
0.1 -> cred_weight(5);
syscreate("credwt",1,"line") -> filed;
cred_weight -> datafile(filed);
sysclose(filed);

pr('histories of decisions for each expert');nl(1);
newarray([1 ^number_of_experts]) -> hist_of_decisions;
[] -> hist_of_decisions(1);
[] -> hist_of_decisions(2);
[] -> hist_of_decisions(3);
[] -> hist_of_decisions(4);
[] -> hist_of_decisions(5);
syscreate("histdec",1,"line") -> filed;
hist_of_decisions -> datafile(filed);
sysclose(filed);

pr('history of increments and decrements to credibility weights');nl(1);
newarray([1 ^number_of_experts]) -> hist_of_incs_decs;
[] -> hist_of_incs_decs(1);
[] -> hist_of_incs_decs(2);
[] -> hist_of_incs_decs(3);
[] -> hist_of_incs_decs(4);
[] -> hist_of_incs_decs(5);
syscreate("histid",1,"line") -> filed;
hist_of_incs_decs -> datafile(filed);
sysclose(filed);

pr('parameters used to update credibility weights');nl(3);
;;; PARAMETERS USED TO UPDATE CREDIBILITY WEIGHTS
75 -> percent_for_success;
syscreate("pfors",1,"line") -> filed;
percent_for_success -> datafile(filed);
sysclose(filed);

20 -> percent_for_failure;
syscreate("pforf",1,"line") -> filed;
percent_for_failure -> datafile(filed);
sysclose(filed);
```

```
20.0 -> N;
syscreate("mathn",1,"line") -> filed;
N -> datafile(filed);
sysclose(filed);

1.0 -> ro;
syscreate("mathro",1,"line") -> filed;
ro -> datafile(filed);
sysclose(filed);

20.0 -> kappa;
syscreate("mathk",1,"line") -> filed;
kappa -> datafile(filed);
sysclose(filed);

;;; ---------------------------------------------------
;;;
;;; SOME MORE CONTROL VARIABLES
;;; ---------------------------------------------------

pr('SOME MORE CONTROL VARIABLES');nl(2);

pr('history of calculated values of decisions');nl(1);
[] -> history_of_real_decision_values;
syscreate("histrv",1,"line") -> filed;
history_of_real_decision_values -> datafile(filed);
sysclose(filed);

newarray([1 ^number_of_experts]) -> array_of_lists;
1 -> i;
until i > number_of_experts
do
  [] -> array_of_lists(i);
  i + 1 -> i
enduntil;
pr('history of parameter values used in changing credibility weights');nl(1);
array_of_lists -> p_values;
array_of_lists -> q_values;
array_of_lists -> f_values;
array_of_lists -> inc_values;
syscreate("pfunc",1,"line") -> filed;
p_values -> datafile(filed);
sysclose(filed);
syscreate("qfunc",1,"line") -> filed;
q_values -> datafile(filed);
sysclose(filed);
syscreate("ffunc",1,"line") -> filed;
f_values -> datafile(filed);
sysclose(filed);
syscreate("incvals",1,"line") -> filed;
inc_values -> datafile(filed);
sysclose(filed);

pr('thresholds used to adapt the fuzzy system');nl(3);
0.85 -> positive_threshold;
syscreate("pthresh",1,"line") -> filed;
positive_threshold -> datafile(filed);
sysclose(filed);

;;; ---------------------------------------------------
;;;
;;; VARIABLES USED IN INDUCTIVE LEARNING FROM EXAMPLES
;;; ---------------------------------------------------

pr('VARIABLES USED IN INDUCTIVE LEARNING FROM
        EXAMPLES');nl(2);
pr('example sets');nl(1);
pr('tally of occurreaces of consequents');nl(1);
;;; SUBSTRATE ONLY STRUCTURE
newarray([1 7]) -> consequent_tally;
1 -> i;
until i > 7
do
  0 -> consequent_tally(i);
  i + 1 -> i
enduntil;
syscreate("conrec1",1,"line") -> filed;
consequent_tally -> datafile(filed);
sysclose(filed);
[] -> example_set_fuzzy_premise;
syscreate("pexset1",1,"line") -> filed;
example_set_fuzzy_premise -> datafile(filed);
sysclose(filed);
[] -> example_set_boolean_premise;
syscreate("bexset1",1,"line") -> filed;
example_set_boolean_premise -> datafile(filed);
sysclose(filed);
[] -> example_set_consequent;
syscreate("cexset1",1,"line") -> filed;
example_set_consequent -> datafile(filed);
sysclose(filed);
newarray([1 7]) -> consequents_in_rules;
5 -> consequents_in_rules(1);
5 -> consequents_in_rules(2);
1 -> consequents_in_rules(3);
1 -> consequents_in_rules(4);
1 -> consequents_in_rules(5);
5 -> consequents_in_rules(6);
1 -> consequents_in_rules(7);
syscreate("cinr1",1,"line") -> filed;
consequents_in_rules -> datafile(filed);
sysclose(filed);

;;; SINGLE LAYER STRUCTURE
newarray([1 18]) -> consequent_tally;
1 -> i;
until i > 18
do
  0 -> consequent_tally(i);
  i + 1 -> i
enduntil;
syscreate("conrec2",1,"line") -> filed;
consequent_tally -> datafile(filed);
sysclose(filed);
[] -> example_set_fuzzy_premise;
syscreate("pexset2",1,"line") -> filed;
example_set_fuzzy_premise -> datafile(filed);
sysclose(filed);
[] -> example_set_boolean_premise;
syscreate("bexset2",1,"line") -> filed;
example_set_boolean_premise -> datafile(filed);
sysclose(filed);
[] -> example_set_consequent;
syscreate("cexset2",1,"line") -> filed;
example_set_consequent -> datafile(filed);
sysclose(filed);
newarray([1 18]) -> consequents_in_rules;
1 -> consequents_in_rules(1);
1 -> consequents_in_rules(2);
6 -> consequents_in_rules(3);
5 -> consequents_in_rules(4);
32 -> consequents_in_rules(5);
31 -> consequents_in_rules(6);
1 -> consequents_in_rules(7);
5 -> consequents_in_rules(8);
7 -> consequents_in_rules(9);
12 -> consequents_in_rules(10);
2 -> consequents_in_rules(11);
2 -> consequents_in_rules(12);
1 -> consequents_in_rules(13);
2 -> consequents_in_rules(14);
6 -> consequents_in_rules(15);
1 -> consequents_in_rules(16);
1 -> consequents_in_rules(17);
1 -> consequents_in_rules(18);
syscreate("cinr2",1,"line") -> filed;
consequents_in_rules -> datafile(filed);
sysclose(filed);

;;; MULTIPLE LAYERS STRUCTURE
newarray([1 7]) -> consequent_tally;
1 -> i;
until i > 7
do
  0 -> consequent_tally(i);
  i + 1 -> i
enduntil;
syscreate("conrec3",1,"line") -> filed;
consequent_tally -> datafile(filed);
sysclose(filed);
[] -> example_set_fuzzy_premise;
syscreate("pexset3",1,"line") -> filed;
example_set_fuzzy_premise -> datafile(filed);
sysclose(filed);
[] -> example_set_boolean_premise;
syscreate("bexset3",1,"line") -> filed;
example_set_boolean_premise -> datafile(filed);
sysclose(filed);
```

```
    [] -> example_set_consequent;
    syscreate("cexset3",1,"line") -> filed;
    example_set_consequent -> datafile(filed);
    sysclose(filed);
    newarray([1 10]) -> consequents_in_rules;
    0 -> consequents_in_rules(1);
    5 -> consequents_in_rules(2);
    5 -> consequents_in_rules(3);
    6 -> consequents_in_rules(4);
    1 -> consequents_in_rules(5);
    6 -> consequents_in_rules(6);
    1 -> consequents_in_rules(7);
    1 -> consequents_in_rules(8);
    1 -> consequents_in_rules(9);
    5 -> consequents_in_rules(10);
    syscreate("cinr3",1,"line") -> filed;
    consequents_in_rules -> datafile(filed);
    sysclose(filed);

    ;;; MQW AND SUPERLATTICE STRUCTURES
    newarray([1 14]) -> consequent_tally;
    1 -> i;
    until i > 14
    do
      0 -> consequent_tally(i);
      i + 1 -> i
    enduntil;
    syscreate("conrcc4",1,"line") -> filed;
    consequent_tally -> datafile(filed);
    sysclose(filed);
    [] -> example_set_fuzzy_premise;
    syscreate("pexset4",1,"line") -> filed;
    example_set_fuzzy_premise -> datafile(filed);
    sysclose(filed);
    [] -> example_set_boolean_premise;
    syscreate("bexset4",1,"line") -> filed;
    example_set_boolean_premise -> datafile(filed);
    sysclose(filed);
    [] -> example_set_consequent;
    syscreate("cexset4",1,"line") -> filed;
    example_set_consequent -> datafile(filed);
    sysclose(filed);
    newarray([1 14]) -> consequents_in_rules;
    3 -> consequents_in_rules(1);
    25 -> consequents_in_rules(2);
    5 -> consequents_in_rules(3);
    1 -> consequents_in_rules(4);
    5 -> consequents_in_rules(5);
    5 -> consequents_in_rules(6);
    10 -> consequents_in_rules(7);
    7 -> consequents_in_rules(8);
    5 -> consequents_in_rules(9);
    1 -> consequents_in_rules(10);
    5 -> consequents_in_rules(11);
    1 -> consequents_in_rules(12);
    2 -> consequents_in_rules(13);
    1 -> consequents_in_rules(14);
    syscreate("cinr4",1,"line") -> filed;
    consequents_in_rules -> datafile(filed);
    sysclose(filed);

  enddefmethod;

endflavour;
```

[2] The program 'fuzzy.p' is the main program that runs the fuzzy system.

```
;;; Program Name: fuzzy.p

;;; Main program for running the fuzzy system

;;;   INDEX OF OBJECTS

;;; 1. Slot Object
;;; 2. Question-Slot Object
;;; 3. Yes-or-No-Question-Slot Object
;;; 4. Procedure-Slot Object
;;; 5. Some important functions

;;; 6. Frame Object
;;; 7. Leading-Frame Object
;;; 8. Experiment-Frame Object
;;; 9. Layer-Frame Object

;;; 10. Substrate-Peak-Frame Object
;;; 11. Layer-Peak-Frame Object
;;; 12. Multiple-layer-peaks-frame Object
;;; 13. Single-Satellite-Peak-Frame Object
;;; 14. Satellite-Peaks-Frame Object
;;; 15. Zero-Order_Peak_Frame Object
;;; 16. Interference-Fringes-Frame Object

;;; 17. Substrate-Only-Frame Object
;;; 18. Single-Layer-Frame Object
;;; 19. A-Number-of-Layers-Frame Object
;;; 20. Multiple-Layers-Sub-Frame Object
;;; 21. Block-of-Layers-Frame Object
;;; 22. Superlattice-Frame Object
;;; 23. Multiple-Layers-Frame Object
;;; 24. MQW-Structure-Frame
;;; 25. Superlattice-With-a-Few-Layers-Frame Object
;;; 26. Frame-System Object
;;; 27. Structural-Parameter-Frame Object

;;; 28. Fuzzy-Variable Object
;;; 29. Rule Object
;;; 30. Function to add rules to a Connection Matrix
;;;   (called add_rules_to_CM())
;;; 31. Connection-Matrix Object
;;; 32. Combined-Matrix Object
;;; 33. Function to check if element is a member of a list
;;; 34. Fuzzy-System Object


;;;   DECLARE GLOBAL VARIABLES

;;; CONTROL VARIABLE: x=1 for experimental curve; x=2 for simulated
curve
vars x;
0 -> x;

vars i j angst B filed;
vars number_of_rules rule_history masking_of_rule N;
vars number_of_fuzzy_variables variable_history fuzzy_rules rule_variable;
vars number_of_experts cred_weight an_expert exp_variable;
vars rule hist_of_decisions hist_of_incs_decs;
vars list_of_rules_used list_of_experts thickness var_changes;
vars positive_threshold;

;;; GLOBAL VARIABLES extracted from frames AND NOT USED IN RULES

;;; From Leading Frame
vars type_of_structure type_of_structure2 grading;

;;; From Experiment Frame 1
vars wavelength h_reflection_index k_reflection_index l_reflection_index;
vars substrate_material reflection_orientation;
vars h_surface_normal_index k_surface_normal_index l_surface_normal_index;
vars Bragg_angle start_of_scan_range end_of_scan_range lattice_parameter;
vars Poisson_ratio spacing_of_planes mat1 mat2;

;;; From Layer-Frame
vars layer_material layer_chemistry layer_thickness;

;;; From Substrate-Peak Frame
vars substrate_FWHM substrate_integrated_intensity_of_peak ;
vars split_substrate_peak;

;;; From Layer-Peak Frame
vars layer_FWHM layer_peak_broadening layer_integrated_intensity_of_peak;
vars layer_split_peak;

;;; From Multiple-Layer-Peaks Frame
vars multiple_layers_peak_broadening multiple_layers_any_asymmetry;
vars multiple_layers_any_wedge_shaped_peaks;

;;; From Single-Satellite-Peak Frame
vars satellite_intensity_of_peak satellite_FWHM;
vars satellite_order satellite_position;

;;; From Satellite-Peaks Frame
vars satellite_spacing_of_peaks separation_of_satellite_peaks;
vars satellite_number_of_peaks satellite_relative_intensities;
vars satellite_relative_width_of_peaks;
vars satellite_relative_integrated_intensities;
vars satellite_subsidiary_interference_effects;
vars satellite_peak_splitting satellite_broadening_of_higher_order_peaks;
vars satellite_order_1 satellite_position_1 satellite_order_2 satellite_position_2;

;;; From Zero-Order Peak Frame
vars zero_order_FWHM zero_order_peak_broadening;
vars zero_order_integrated_intensity_of_peak;

;;; From Interference-Fringes_frame
vars spacing_of_interference_fringes;

;;; From Substrate-Only Frame
vars number_of_layers number_of_peaks;

;;; From Single-Layer-Frame
vars layer_frame substrate_layer_peak_overlap;
vars peak_splitting relaxed_layer three_times_width_of_peak;

;;; From A-Number-of-Layers-Frame
vars layer_composition number_of_blocks layer_frame;

;;; From Superlattice-Frame
vars thickness_of_superlattice more_than_two_peaks identify_the_peaks;

;;; From Multiple-Layers Frame
vars layers_in_blocks;

;;; From Superlattice-With-a-Few_Layers Frame
vars top_layer_composition top_layer_thickness bottom_layer_composition;
vars bottom_layer_thickness number_of_top_layers number_of_bottom_layers;
vars overlap_of_peaks;

;;; BOOLEAN VARIABLES used in rules

vars type_of_structure_is_substrate_only number_of_peaks_is_one;
vars number_of_peaks_is_more_than_one;
vars number_of_peaks_is_none;

vars type_of_structure_is_single_layer number_of_peaks_is_two;
vars number_of_peaks_is_more_than_two;
vars substrate_material_equal_to_layer peak_separation_is_low;
vars layer_integrated_intensity_of_peak_is_zero;
vars layer_thickness_greater_than_half_micron;
vars layer_thickness_less_than_5_microns peak_splitting_is_zero;
vars peak_splitting_less_than_three_times_width_of_peak;
vars relaxed_mismatch_is_high peak_splitting_is_high;
vars spacing_of_interference_fringes_is_low;

vars type_of_structure_is_multiple_layers;
vars correspondence_between_layers_and_peaks;

vars type_of_structure_is_MQW satellite_spacing_greater_than_zero;
vars satellite_relative_width_of_peaks_greater_than_zero;
vars satellite_relative_integrated_intensities_greater_than_zero;
vars thickness_of_superlattice_less_than_half_micron;
vars type_of_structure_has_additional_layers;


;;; FUZZY VARIABLES used in the premises of rules

vars substrate_asymmetry_in_peak substrate_peak_broadening;
vars layer_asymmetry_in_peak layer_wedge_shaped_peak;
vars intensity_of_layer_peak;
```

```
vars multiple_layers_asymmetry_in_peak multiple_layers_wedge_shaped_peak;
vars satellite_visibility satellite_asymmetry_of_plus_and_minus_peaks;
vars zero_order_asymmetry_in_peak interference_fringes;
vars visibility_of_interference_fringes;


;;; VARIABLES USED IN MATHEMATICAL FORMULA FOR CHANGING
            CREDIBILITY WEIGHTS

vars ro kappa original_list_of_rules;

;;; VARIABLES USED TO STORE FUZZY SETS

;;; ANTECEDENTS

vars fuzzy_peak_broadening;
[0.0 0.15 0.1 0.4 0.3 0.6 0.5 0.8 0.7 1.0] -> fuzzy_peak_broadening;
vars fuzzy_asymmetry;
[0.0 0.1 0.5 0.2 0.1 0.5 0.4 0.8 0.7 1.0] -> fuzzy_asymmetry;
vars fuzzy_wedge_shaped;
[0.0 0.1 0.5 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_wedge_shaped;
vars fuzzy_visibility;
[0.0 0.1 0.5 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_visibility;
vars fuzzy_spacing_of_peaks;
[0.0 0.1 0.5 0.3 0.2 0.5 0.4 0.7 0.6 1.0] -> fuzzy_spacing_of_peaks;
vars fuzzy_plus_minus_asymmetry;
[0.0 0.1 0.5 0.2 0.1 0.5 0.4 0.8 0.7 1.0] -> fuzzy_plus_minus_asymmetry;
vars fuzzy_interference;
[0.0 0.1 0.5 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_interference;
vars fuzzy_intensity_of_peak;
vars fuzzy_evidence_of_mismatch;
vars fuzzy_crystal_quality;
vars fuzzy_grading_of_the_layer;
vars fuzzy_layer_is_thick;
vars fuzzy_layer_is_thin;

vars membership_functions mf;

;;; CONSEQUENTS

vars fuzzy_variable;

vars dummy;
0 -> dummy;

;;; VARIABLES USED FOR CALCULATIONS
vars experimental_mismatch relaxed_mismatch;
vars period_of_superlattice period_dispersion thickness_of_layer;

;;; SOME STRUCTURAL PARAMETERS
;;; (some are taken from above, others are derived in
;;;  the Structural-Parameter-Frame Object below)
;;;
;;; Parameters deduced from Rules
vars crystal_quality strain_in_surface_layer_of_sample;
vars reference_crystal_is_different_to_substrate incorrect_experiment;
vars characteristic_curve;
vars bending_of_substrate grading_of_the_layer evidence_of_mismatch;
vars change_in_lattice_parameter_with_depth;
vars layer_is_present_in_the_substrate_peak;
vars layer_is_thin misoriented_or_mismatched_layer;
vars simulation_or_calibration_chart_is_needed;
vars layer_is_thick layers_are_thick there_are_hidden_layers_somewhere;
vars layers_are_thin;
vars grading_occurs_through_AB_layers;
vars grading_or_dispersion_of_layer_thicknesses;
vars relaxation misorientation_of_substrate peak_is_outside_the_scan_range;
vars grading_distance;
vars multiple_layers large_overall_layer_thickness layers_are_not_uniform;
vars crystal_consists_of_subgrains;
vars structural_parameter there_are_thin_layers_at_the_interfaces;
vars evidence_of_interferometer_structure;

;;; SPECIAL VARIABLES USED TO RECORD CHANGES
vars p_value q_value f_value inc_value;

;;; VARIABLES USED TO FIRE RULES
vars existing_rules dummy_array consequences layers fp bp;
vars rule_list temp_list count value Boolean number_of_variables;
vars list1 list2 dummy2 k fuzzy mem membership;

;;; ____________________________________________________
```

```
;;;    1. SLOT OBJECT

flavour SLOT;
   ivars name value;

      ;;; Fill a named slot with a value
      defmethod fill_slot (name_of_slot, value_of_slot);
   lvars name_of_slot value_of_slot;

         name_of_slot -> name;
         value_of_slot -> value;
      enddefmethod;

      ;;; Print the name and value of a slot on the screen
      defmethod printself;
         pr('The value of slot ');pr(name);pr(' is ');ppr(value);pr('.')
      enddefmethod;

endflavour;


;;;    2. QUESTION-SLOT OBJECT

flavour QUESTION_SLOT isa SLOT;

   ;;; Fill the named slot with a value typed in as the answer to a question
   defmethod fill_slot (name_of_slot, quest);
 lvars name_of_slot quest;

      name_of_slot -> name;
      ppr(quest);
      readline() -> value;
   enddefmethod;

   ;;; All other methods are INHERITED from the SLOT object

endflavour;

;;;    3. YES_OR_NO_QUESTION_SLOT OBJECT

flavour Y_OR_N_QUESTION_SLOT isa SLOT;

   ;;; Fill the named slot with a Boolean value typed in as the answer to a
   ;;; question.
   defmethod fill_slot (name_of_slot, quest);
 lvars name_of_slot quest;

      name_of_slot -> name;
      ppr(quest);
      readline() -> value;
 if not(value=[y] or value=[Y] or value=[n] or value=[N])
 then
   pr('_____');nl(1);
   pr('You must type Y or N in answer to this question.');nl(1);
   pr('Please try again ...');nl(1);
   pr('_____');nl(1);
   fill_slot(name_of_slot, quest)
 endif;
   enddefmethod;

   ;;; All other methods are INHERITED from the SLOT object

endflavour;


;;; 4. PROCEDURE_SLOT OBJECT

flavour PROCEDURE_SLOT isa SLOT;

 defmethod fill_slot(name_of_slot, proc_or_func);
   lvars name_of_slot proc_or_func;

   name_of_slot -> name;
   apply(proc_or_func) -> value
 enddefmethod;

endflavour;
```

```
;;; 5. SOME IMPORTANT FUNCTIONS

define verify_input(x);
 lvars x, check;

      if (x=[a] or x=[A] or x=[b] or x=[B] or x=[c] or x=[C] or x=[d] or
x=[D] or x=[e] or x=[E])
       then
          1 -> check
      else
          nl(4);pr('You have typed an incorrect answer to this question ...');
          nl(2);pr('Please, try again ...');nl(1);
   0 -> check
  endif;
  check
enddefine;


define assign_fuzzy_value(x, fuzzy_set);
 lvars x fuzzy_set fuzzy_value;

  ;;; ASSIGN FUZZY-SET-VALUE TO THE VARIABLE
  if (x=[a] or x=[A])
  then
    (fuzzy_set(10) + fuzzy_set(9)) / 2 -> fuzzy_value
  elseif (x=[b] or x=[B])
  then
    (fuzzy_set(8) + fuzzy_set(7)) / 2 -> fuzzy_value
  elseif (x=[c] or x=[C])
  then
    (fuzzy_set(6) + fuzzy_set(5)) / 2 -> fuzzy_value
  elseif (x=[d] or x=[D])
  then
    (fuzzy_set(4) + fuzzy_set(3)) / 2 -> fuzzy_value
  else
    (fuzzy_set(2) + fuzzy_set(1)) / 2 -> fuzzy_value
  endif;
  fuzzy_value
enddefine;

define assign_upper_case(x);
 lvars x;

 if x=[y]
 then
   [Y] -> x
 endif;

 if x=[n]
 then
   [N] -> x
 endif;

 x
enddefine;

define test_integer(x);
 lvars x check;

 if islist(x)
 then
   if x=[]
   then
   else
     hd(x) -> x
   endif;
 endif;

   if datakey(x)=integer_key or datakey(x)=biginteger_key
   then
     1 -> check
   else
     nl(4);pr('You have typed an incorrect answer to this question ...');
      nl(2);pr('Please, try again ...');nl(1);
     0 -> check
   endif;
 check
enddefine;

define test_number(x);
 lvars x check;

 if islist(x)
 then
   if x=[]
   then
   else
     hd(x) -> x
   endif;
 endif;

     if datakey(x)=integer_key or datakey(x)=biginteger_key or
          datakey(x)=decimal_key or datakey(x)=ddecimal_key
     then
        1 -> check
     else
        nl(4);pr('You have typed an incorrect answer to this question ...');
        nl(2);pr('Please, try again ...');nl(1);
   0 -> check
 endif;
 check
enddefine;


define test_string(x);
 lvars x check;

 if islist(x)
 then
   if x=[]
   then
   else
     hd(x) -> x
   endif;
 endif;

     if (datakey(x)=string_key or datakey(x)=word_key)
      then
        1 -> check
     else
        nl(4);pr('You have typed an incorrect answer to this question ...');
        nl(2);pr('Please, try again ...');nl(1);
   0 -> check
 endif;
 check
enddefine;


define clean(x);
 lvars x;

 (1000.0 * x) -> x;
 round(x) -> x;
 (x / 1000.0) -> x;
 x
enddefine;

define test_for_broadening(FWHM);
 lvars Si Ge AlP AlAs AlSb GaP GaAs GaSb InP InAs InSb wave;
 lvars FWHM check i j materials width wave1 wave2;

 hd(FWHM) -> FWHM;

 ;;; Types of materials
 newarray([1 11]) -> materials;
 'Si' -> materials(1);
 'Ge' -> materials(2);
 'AlP' -> materials(3);
 'AlAs' -> materials(4);
 'AlSb' -> materials(5);
 'GaP' -> materials(6);
 'GaAs' -> materials(7);
 'GaSb' -> materials(8);
 'InP' -> materials(9);
 'InAs' -> materials(10);
 'InSb' -> materials(11);

 ;;; FWHMs of substrate peaks for these materials
 [8.4 7.6 3.9 2.4 4.8] -> Si;
 [17.1 15.5 7.4 4.8 2.9] -> Ge;
 [8.0 7.5 4.0 2.5 4.8] -> AlP;
 [12.4 11.8 6.0 2.6 2.4] -> AlAs;
 [17.6 12.7 7.9 3.2 2.5] -> AlSb;
 [13.8 12.9 6.4 2.6 2.4] -> GaP;
 [17.0 15.5 7.4 4.8 2.9] -> GaAs;
 [27.1 21.8 11.4 5.7 4.8] -> GaSb;
 [19.8 14.5 8.7 4.0 2.5] -> InP;
```

```
[24.3 16.2 10.1 5.1 3.6] -> InAs;
[26.0 17.1 11.9 5.4 4.8] -> InSb;

;;; Wavelengths with values stored for FWHMs
[2.29 1.94 1.54 0.71 0.56] -> wave;

0 -> check;
;;; Only values for symmetric reflections are stored using 0 0 4 Miller indices
if (h_reflection_index=0 and k_reflection_index=0 and l_reflection_index=4
    and h_surface_normal_index=0 and k_surface_normal_index=0 and
    l_surface_normal_index=1 and reflection_orientation='Symmetric')
then
 1 -> i;
 until i > 5
 do
  wave(i) - 0.01 -> wave1;
  wave(i) + 0.01 -> wave2;
  if (wavelength > wave1 and wavelength < wave2)
  then
   1 -> j;
   until j > 11
   do
    if substrate_material = materials(j)
    then
     if j=1
     then
      Si(i) -> width;
     elseif j=2
     then
      Ge(i) -> width;
     elseif j=2
     then
      Ge(i) -> width;
     elseif j=3
     then
      AlP(i) -> width;
     elseif j=4
     then
      AlAs(i) -> width;
     elseif j=5
     then
      AlSb(i) -> width;
     elseif j=6
     then
      GaP(i) -> width;
     elseif j=7
     then
      GaAs(i) -> width;
     elseif j=8
     then
      GaSb(i) -> width;
     elseif j=9
     then
      InP(i) -> width;
     elseif j=10
     then
      InAs(i) -> width;
     else
      InSb(i) -> width;
     endif;

     if FWHM > 2 * width
     then
      assign_fuzzy_value([a],fuzzy_peak_broadening)
           -> substrate_peak_broadening;
     elseif FWHM > width + (width * 50/100)
     then
      assign_fuzzy_value([b],fuzzy_peak_broadening)
           -> substrate_peak_broadening;
     elseif FWHM > width + (width * 25/100)
     then
      assign_fuzzy_value([c],fuzzy_peak_broadening)
           -> substrate_peak_broadening;
     elseif FWHM > width + (width * 10/100)
     then
      assign_fuzzy_value([d],fuzzy_peak_broadening)
           -> substrate_peak_broadening;
     else
      assign_fuzzy_value([e],fuzzy_peak_broadening)
           -> substrate_peak_broadening;
     endif;

     1 -> check;
    endif;
      j+1 -> j;
    enduntil;
   endif;
   i+1 -> i;
  enduntil;
 endif;

 if check = 0
 then
  pr('Broadening of the substrate peak is automatically calculated');nl(1);
  pr(' for the 0 0 4 symmetric reflection in the cases when the ');nl(1);
  pr(' wavelength of the radiation source is either 2.29 Angstroms');nl(1);
  pr(' or 1.94 Angstroms or 1.54 Angstroms or 0.71 Angstroms ');nl(1);
  pr(' or 0.56 Angstroms. In all other cases the user must decide ');nl(1);
  pr(' whether there is broadening of the substrate peak.');nl(3);
 endif;

 check
enddefine;

;;; ____________________________________________

;;;    6. FRAME OBJECT

flavour FRAME;
  ivars name_of_frame number_of_slots;

  ;;; Initialise a frame by giving it a name & number of slots
  defmethod initialise_the_frame(name,number);
 lvars name number;

     name -> name_of_frame;
     number -> number_of_slots
  enddefmethod;

  ;;; Print the name of the frame and no. of slots on the screen
  defmethod printself;
     pr('The');pr(name_of_frame);pr('has');pr(number_of_slots);pr('slots.')
  enddefmethod;

endflavour;

;;;_____________________________________________

;;;    7. LEADING_FRAME OBJECT

flavour LEADING_FRAME isa FRAME;
 ivars slot1 slot2;

 ;;; This method allows you to fill the LEADING_FRAME
 defmethod fill_the_frame;
  lvars check quest name_of_slot2;

  "LEADING_FRAME" -> name_of_frame;
  2 -> number_of_slots;

  ;;; check will stay =0 until correct values are entered
  0 -> check;
  while (check=0)
  do
   ;;;    Screen display for the LEADING_FRAME
   nl(10);
   pr('          LEADING QUESTION ');nl(3);
   pr('Which of the following best describes the structure?');nl(2);
   pr('   (a) substrate only');nl(1);
   pr('   (b) a single layer');nl(1);
   pr('   (c) MQW structure');nl(1);
   pr('   (d) non-graded multiple layers');nl(1);
   pr('   (e) superlattice capped with a few layers top and bottom;');nl(1);
   pr('   (f) a single graded layer');nl(1);
   pr('   (g) graded multiple layers');nl(1);
   pr('   (h) graded superlattice.');nl(2);

   ;;;    Assign leading question to a variable: quest
   'Type the letter of your choice ...'->quest;

   make_instance([QUESTION_SLOT])->slot1;
   slot1 <- fill_slot("TYPE_OF_STRUCTURE",quest);

   make_instance([SLOT])->slot2;
   "STRUCTURE_IS"->name_of_slot2;
```

```
    1 -> check;
    if (slot1 <-value=[a] or slot1 <-value=[A])
    then
      slot2 <- fill_slot(name_of_slot2,'substrate only');
      1 -> type_of_structure_is_substrate_only;
      0.0 -> grading;
    elseif (slot1 <-value=[b] or slot1 <-value=[B])
    then
      slot2 <- fill_slot(name_of_slot2,'a single layer');
      1 -> type_of_structure_is_single_layer;
      0.0 -> grading;
    elseif (slot1 <-value=[f] or slot1 <-value=[F])
    then
      slot2 <- fill_slot(name_of_slot2,'a single layer');
      1 -> type_of_structure_is_single_layer;
      1.0 -> grading;
    elseif (slot1 <-value=[d] or slot1 <-value=[D])
    then
      slot2 <- fill_slot(name_of_slot2,'multiple layers');
      1 -> type_of_structure_is_multiple_layers;
      0.0 -> grading;
    elseif (slot1 <-value=[g] or slot1 <-value=[G])
    then
      slot2 <- fill_slot(name_of_slot2,'multiple layers');
      1 -> type_of_structure_is_multiple_layers;
      1.0 -> grading;
    elseif (slot1 <-value=[c] or slot1 <-value=[C])
    then
      slot2 <- fill_slot(name_of_slot2,'MQW structure');
      1 -> type_of_structure_is_MQW;
      0.0 -> grading;
    elseif (slot1 <-value=[e] or slot1 <-value=[E])
    then
      slot2 <- fill_slot(name_of_slot2,'superlattice capped with a few layers top
               and bottom');
      'additional layers' -> type_of_structure2;
      1 -> type_of_structure_is_MQW;
      1 -> type_of_structure_has_additional_layers;
      0.0 -> grading;
    elseif (slot1 <-value=[h] or slot1 <-value=[H])
    then
      slot2 <- fill_slot(name_of_slot2,'superlattice');
      1 -> type_of_structure_is_MQW;
      1.0 -> grading;
    else
      nl(4);pr('You have typed an incorrect answer to this question ...');
      nl(2);pr('Please, try again ...');nl(1);
      0 -> check
    endif;
  endwhile;

 enddefmethod;


 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  slot2<-value -> type_of_structure
 enddefmethod;

endflavour;


;;;    8. EXPERIMENT_FRAME OBJECT

flavour EXPERIMENT_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5 slot6 slot7 slot8 slot9 slot10;
 ivars slot11 slot12 slot13 slot14 slot15 slot16 slot17;

 ;;; This method allows you to fill the EXPERIMENT_FRAME
 defmethod fill_the_frame;
 lvars check name_of_slot6 name_of_slot8 name_of_slot15 name_of_slot16;
 lvars quest1 quest2 quest3 quest4 quest5 quest7 quest9 quest10 quest11 quest13
quest14;
 lvars h k l a d lambda theta;

 "EXPERIMENT_INFORMATION_FRAME" -> name_of_frame;
 17 -> number_of_slots;

 ;;;    Screen display for the EXPERIMENT_FRAME
 nl(3);
 pr('          GENERAL INFORMATION ON THE EXPERIMENT');nl(3);

 ;;; SLOT 1

 ;;; Create slot1 of the EXPERIMENT_FRAME
 make_instance([QUESTION_SLOT])->slot1;
 '1. What is the Wavelength of the X-ray beam (in angstroms)' -> quest1;

 0 -> check;
 while (check=0)
 do
  ;;; Fill slot1 of the EXPERIMENT_FRAME
  slot1 <- fill_slot("WAVELENGTH",quest1);nl(3);
  test_number(slot1<-value) -> check;
 endwhile;

 ;;; SECOND QUESTION: FILLS SLOTS 2,3 &4

 pr('2. Please enter the reflection indices for this experiment,');nl(1);
 pr('  i.e. the h,k,l Miller indices multiplied by the order of diffraction:');nl(2);

 make_instance([QUESTION_SLOT])->slot2;

 '     H: ' -> quest2;

 0 -> check;
 while (check=0)
 do
  slot2 <- fill_slot("H_REFLECTION_INDEX",quest2);nl(2);
  test_integer(slot2<-valuc) -> check;
 endwhile;

 make_instance([QUESTION_SLOT])->slot3;

 '     K: ' -> quest3;

 0 -> check;
 while (check=0)
 do
  slot3 <- fill_slot("K_REFLECTION_INDEX",quest3);nl(2);
  test_integer(slot3<-value) -> check;
 endwhile;

 make_instance([QUESTION_SLOT])->slot4;

 '     L: ' -> quest4;

 0 -> check;
 while (check=0)
 do
  slot4 <- fill_slot("H_RELECTION_INDEX",quest4);nl(3);
  test_integer(slot4<-value) -> check;
 endwhile;

 0 -> check;
 while (check=0)
 do

  ;;; QUESTION THREE: FILLS SLOTS 5 & 6

  pr('3. Which of the following is the substrate material?');nl(2);
  pr('    (a) Si');nl(1);
  pr('    (b) Ge');nl(1);
  pr('    (c) AlP');nl(1);
  pr('    (d) AlAs');nl(1);
  pr('    (e) AlSb');nl(1);
  pr('    (f) GaP');nl(1);
  pr('    (g) GaAs');nl(1);
  pr('    (h) GaSb');nl(1);
  pr('    (i) InP');nl(1);
  pr('    (j) InAs');nl(1);
  pr('    (k) InSb');nl(2);

  'Type the letter of your choice ...'->quest5;
  make_instance([QUESTION_SLOT])->slot5;
  slot5 <- fill_slot("SUBSTRATE_MATERIAL",quest5);

  make_instance([SLOT])->slot6;
  "SUBSTRATE_IS"->name_of_slot6;
  make_instance([SLOT])->slot15;
  "LATTICE_PARAMETER"->name_of_slot15;
  make_instance([SLOT])->slot16;
  "POISSON_RATIO"->name_of_slot16;

  1 -> check;
  if (slot5<-value=[a] or slot5<-value=[A])
  then
    slot6 <- fill_slot(name_of_slot6,'Si');
```

```
    slot15 <- fill_slot(name_of_slot15, 5.4309);
    slot16 <- fill_slot(name_of_slot16, 0.278);
    1 -> mat1;
  elseif (slot5<-value=[b] or slot5<-value=[B])
  then
    slot6 <- fill_slot(name_of_slot6,'Ge');
    slot15 <- fill_slot(name_of_slot15, 5.6461);
    slot16 <- fill_slot(name_of_slot16, 0.271);
    2 -> mat1;
  elseif (slot5<-value=[c] or slot5<-value=[C])
  then
    slot6 <- fill_slot(name_of_slot6,'AlP');
    slot15 <- fill_slot(name_of_slot15, 5.451);
    slot16 <- fill_slot(name_of_slot16, 0.28);
    3 -> mat1;
  elseif (slot5<-value=[d] or slot5<-value=[D])
  then
    slot6 <- fill_slot(name_of_slot6,'AlAs');
    slot15 <- fill_slot(name_of_slot15, 5.6623);
    slot16 <- fill_slot(name_of_slot16, 0.28);
    4 -> mat1;
  elseif (slot5<-value=[e] or slot5<-value=[E])
  then
    slot6 <- fill_slot(name_of_slot6,'AlSb');
    slot15 <- fill_slot(name_of_slot15, 6.1355);
    slot16 <- fill_slot(name_of_slot16, 0.317);
    5 -> mat1;
  elseif (slot5<-value=[f] or slot5<-value=[F])
  then
    slot6 <- fill_slot(name_of_slot6,'GaP');
    slot15 <- fill_slot(name_of_slot15, 5.4505);
    slot16 <- fill_slot(name_of_slot16, 0.271);
    6 -> mat1;
  elseif (slot5<-value=[g] or slot5<-value=[G])
  then
    slot6 <- fill_slot(name_of_slot6,'GaAs');
    slot15 <- fill_slot(name_of_slot15, 5.65325);
    slot16 <- fill_slot(name_of_slot16, 0.31);
    7 -> mat1;
  elseif (slot5<-value=[h] or slot5<-value=[H])
  then
    slot6 <- fill_slot(name_of_slot6,'GaSb');
    slot15 <- fill_slot(name_of_slot15, 5.6461);
    slot16 <- fill_slot(name_of_slot16, 0.313);
    8 -> mat1;
  elseif (slot5<-value=[i] or slot5<-value=[I])
  then
    slot6 <- fill_slot(name_of_slot6,'InP');
    slot15 <- fill_slot(name_of_slot15, 5.8688);
    slot16 <- fill_slot(name_of_slot16, 0.36);
    9 -> mat1;
  elseif (slot5<-value=[j] or slot5<-value=[J])
  then
    slot6 <- fill_slot(name_of_slot6,'InAs');
    slot15 <- fill_slot(name_of_slot15, 6.05838);
    slot16 <- fill_slot(name_of_slot16, 0.352);
    10 -> mat1;
  elseif (slot5<-value=[k] or slot5<-value=[K])
  then
    slot6 <- fill_slot(name_of_slot6,'InSb');
    slot15 <- fill_slot(name_of_slot15, 6.4788);
    slot16 <- fill_slot(name_of_slot16, 0.353);
    11 -> mat1;
  else
    nl(4);pr('You have typed an incorrect answer to this question ...');
    nl(2);pr('Please, try again ...');nl(1);
    0 -> check
  endif;

endwhile;nl(3);

0 -> check;
while (check=0)
do

  ;;; QUESTION FOUR: FILLS SLOTS 7 & 8

  pr('4. Which of the following is the reflection orientation?');nl(2);
  pr('    (a) Symmetric');nl(1);
  pr('    (b) Asymmetric with glancing incidence');nl(1);
  pr('    (c) Asymmetric with glancing exit');nl(2);

  'Type the letter of your choice ...'->quest7;
  make_instance([QUESTION_SLOT])->slot7;
  slot7 <- fill_slot("REFLECTION_ORIENTATION",quest7);

  make_instance([SLOT])->slot8;
  "REFLECTION_ORIENTATION_IS"->name_of_slot8;
  1 -> check;
  if (slot7<-value=[a] or slot7<-value=[A])
  then
    slot8 <- fill_slot(name_of_slot8,'Symmetric');
  elseif (slot7<-value=[b] or slot7<-value=[B])
  then
    slot8 <- fill_slot(name_of_slot8,'Asymmetric with glancing incidence');
  elseif (slot7<-value=[c] or slot7<-value=[C])
  then
    slot8 <- fill_slot(name_of_slot8,'Asymmetric with glancing exit');
  else
    nl(4);pr('You have typed an incorrect answer to this question ...');
    nl(2);pr('Please, try again ...');nl(1);
    0 -> check
  endif;

endwhile;nl(3);

;;; FIFTH QUESTION: FILLS SLOTS 9,10 & 11

pr('5. Please enter the surface normal indices for this experiment,');nl(2);

make_instance([QUESTION_SLOT])->slot9;

'    H: ' -> quest9;

0 -> check;
while (check=0)
do
  slot9 <- fill_slot("H_SURFACE_NORMAL_INDEX",quest9);nl(2);
  test_integer(slot9<-value) -> check;
endwhile;

make_instance([QUESTION_SLOT])->slot10;

'    K: ' -> quest10;

0 -> check;
while (check=0)
do
  slot10 <- fill_slot("K_SURFACE_NORMAL_INDEX",quest10);nl(2);
  test_integer(slot10<-value) -> check;
endwhile;

make_instance([QUESTION_SLOT])->slot11;

'    L: ' -> quest11;

0 -> check;
while (check=0)
do
  slot11 <- fill_slot("L_SURFACE_NORMAL_INDEX",quest11);nl(3);
  test_integer(slot11<-value) -> check;
endwhile;


;;; SLOT12

make_instance([SLOT])->slot12;
make_instance([SLOT])->slot17;

;;;    Fill slot12 of the EXPERIMENT_FRAME

hd(slot1<-value) -> lambda;
slot15<-value -> a;
hd(slot2<-value) -> h;
hd(slot3<-value) -> k;
hd(slot4<-value) -> l;

a / sqrt(h**h + k**k + l**l) -> d;
arcsin(lambda / (2 * d)) -> theta;

slot12 <- fill_slot("BRAGG_ANGLE", theta);
slot17 <- fill_slot("SPACING_OF_PLANES", d);

;;; QUESTION 7: SLOTS 13 & 14

pr('6. Please type in the scan range of your rocking curve (in arc secs):');nl(2);
make_instance([QUESTION_SLOT])->slot13;
make_instance([QUESTION_SLOT])->slot14;
```

```
'      Start : ' -> quest13;
'      Finish: ' -> quest14;

;;;    Fill slot13 and slot14 of the EXPERIMENT_FRAME
0 -> check;
while (check=0)
do
 slot13 <- fill_slot("START_OF_SCAN_RANGE",quest13);nl(2);
 test_number(slot13<-value) -> check;
endwhile;

0 -> check;
while (check=0)
do
 slot14 <- fill_slot("END_OF_SCAN_RANGE",quest14);nl(3);
 test_number(slot13<-value) -> check;
endwhile;

enddefmethod;

;;; Transform the frame into a logic-based format
defmethod logic_format;
 hd(slot1<-value) -> wavelength;
 hd(slot2<-value) -> h_reflection_index;
 hd(slot3<-value) -> k_reflection_index;
 hd(slot4<-value) -> l_reflection_index;
 slot6<-value -> substrate_material;
 slot8<-value -> reflection_orientation;
 hd(slot9<-value) -> h_surface_normal_index;
 hd(slot10<-value) -> k_surface_normal_index;
 hd(slot11<-value) -> l_surface_normal_index;
 slot12<-value -> Bragg_angle;
 hd(slot13<-value) -> start_of_scan_range;
 hd(slot14<-value) -> end_of_scan_range;
 slot15<-value -> lattice_parameter;
 slot16<-value -> Poisson_ratio;
 slot17<-value -> spacing_of_planes;
enddefmethod;
endflavour;

;;;    9. LAYER_FRAME OBJECT

flavour LAYER_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4;

 ;;; This method allows you to fill the LAYER_FRAME
 defmethod fill_the_frame;
 lvars check quest1 name_of_slot2 name_of_slot3 quest4;

 "LAYER_FRAME" -> name_of_frame;
 4 -> number_of_slots;

 0 -> check;
 while (check=0)
 do
  ;;;    Screen display for the LAYER_FRAME
  nl(3);
  pr('          DESCRIPTION OF LAYER ');nl(3);

  pr(' (a) Which of the following is the layer material?');nl(2);
  pr('  ENDMEMBERS:');nl(1);
  pr('   (a) Si              (g) GaAs');nl(1);
  pr('   (b) Ge              (h) GaSb');nl(1);
  pr('   (c) AlP             (i) InP');nl(1);
  pr('   (d) AlAs            (j) InAs');nl(1);
  pr('   (e) AlSb            (k) InSb');nl(1);
  pr('   (f) GaP');nl(2);
  pr('  COMPOUNDS:');nl(1);
  pr('   (l) GeSi            (q) InAsP');nl(1);
  pr('   (m) InGaP           (r) GaAsP');nl(1);
  pr('   (n) InGaAs          (s) InGaAlAs');nl(1);
  pr('   (o) InGaSb          (t) GaInAsP');nl(1);
  pr('   (p) AlGaAs');nl(2);

  '  Type the letter of your choice ...'->quest1;
  make_instance([QUESTION_SLOT])->slot1;
  slot1 <- fill_slot("LAYER_MATERIAL",quest1);

  make_instance([SLOT])->slot2;
  make_instance([SLOT])->slot3;
  "LAYER_IS"->name_of_slot2;
  "LAYER_CHEMISTRY"->name_of_slot3;

  1 -> check;

  if (slot1<-value=[a] or slot1<-value=[A])
  then
   slot2 <- fill_slot(name_of_slot2,'Si');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   1 -> mat2;
  elseif (slot1<-value=[b] or slot1<-value=[B])
  then
   slot2 <- fill_slot(name_of_slot2,'Ge');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   2 -> mat2;
  elseif (slot1<-value=[c] or slot1<-value=[C])
  then
   slot2 <- fill_slot(name_of_slot2,'AlP');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   3 -> mat2;
  elseif (slot1<-value=[d] or slot1<-value=[D])
  then
   slot2 <- fill_slot(name_of_slot2,'AlAs');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   4 -> mat2;
  elseif (slot1<-value=[e] or slot1<-value=[E])
  then
   slot2 <- fill_slot(name_of_slot2,'AlSb');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   5 -> mat2;
  elseif (slot1<-value=[f] or slot1<-value=[F])
  then
   slot2 <- fill_slot(name_of_slot2,'GaP');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   6 -> mat2;
  elseif (slot1<-value=[g] or slot1<-value=[G])
  then
   slot2 <- fill_slot(name_of_slot2,'GaAs');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   7 -> mat2;
  elseif (slot1<-value=[h] or slot1<-value=[H])
  then
   slot2 <- fill_slot(name_of_slot2,'GaSb');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   8 -> mat2;
  elseif (slot1<-value=[i] or slot1<-value=[I])
  then
   slot2 <- fill_slot(name_of_slot2,'InP');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   9 -> mat2;
  elseif (slot1<-value=[j] or slot1<-value=[J])
  then
   slot2 <- fill_slot(name_of_slot2,'InAs');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   10 -> mat2;
  elseif (slot1<-value=[k] or slot1<-value=[K])
  then
   slot2 <- fill_slot(name_of_slot2,'InSb');
   slot3 <- fill_slot(name_of_slot3,'endmember');
   11 -> mat2;
  elseif (slot1<-value=[l] or slot1<-value=[L])
  then
   slot2 <- fill_slot(name_of_slot2,'GeSi');
   slot3 <- fill_slot(name_of_slot3,'compound');
   12 -> mat2;
  elseif (slot1<-value=[m] or slot1<-value=[M])
  then
   slot2 <- fill_slot(name_of_slot2,'InGaP');
   slot3 <- fill_slot(name_of_slot3,'compound');
   13 -> mat2;
  elseif (slot1<-value=[n] or slot1<-value=[N])
  then
   slot2 <- fill_slot(name_of_slot2,'InGaAs');
   slot3 <- fill_slot(name_of_slot3,'compound');
   14 -> mat2;
  elseif (slot1<-value=[o] or slot1<-value=[O])
  then
   slot2 <- fill_slot(name_of_slot2,'InGaSb');
   slot3 <- fill_slot(name_of_slot3,'compound');
   15 -> mat2;
  elseif (slot1<-value=[p] or slot1<-value=[P])
  then
   slot2 <- fill_slot(name_of_slot2,'AlGaAs');
   slot3 <- fill_slot(name_of_slot3,'compound');
   16 -> mat2;
  elseif (slot1<-value=[q] or slot1<-value=[Q])
  then
   slot2 <- fill_slot(name_of_slot2,'InAsP');
   slot3 <- fill_slot(name_of_slot3,'compound');
```

```
    17 -> mat2;
   elseif (slot1 <-value=[r] or slot1 <-value=[R])
   then
    slot2 <- fill_slot(name_of_slot2,'GaAsP');
    slot3 <- fill_slot(name_of_slot3,'compound');
    18 -> mat2;
   elseif (slot1 <-value=[s] or slot1 <-value=[S])
   then
    slot2 <- fill_slot(name_of_slot2,'InGaAlAs');
    slot3 <- fill_slot(name_of_slot3,'compound');
    19 -> mat2;
   elseif (slot1 <-value=[t] or slot1 <-value=[T])
   then
    slot2 <- fill_slot(name_of_slot2,'GaInAsP');
    slot3 <- fill_slot(name_of_slot3,'compound');
    20 -> mat2;
   else
    nl(4);pr('You have typed an incorrect answer to this question ...');
    nl(2);pr('Please, try again ...');nl(1);
    0 -> check
   endif;

  endwhile;nl(3);

  ;;; SLOT4

  make_instance([QUESTION_SLOT])->slot4;
  ' (b) What is the estimated thickness of the layer (in microns) ' -> quest4;
  0 -> check;
  while (check=0)
  do
   slot4 <- fill_slot("LAYER_THICKNESS",quest4);nl(1);
   test_number(slot4<-value) -> check;
  endwhile;

  enddefmethod;


  ;;; Transform the frame into a logic-based format
  defmethod logic_format;
   slot2<-value -> layer_material;
   slot3<-value -> layer_chemistry;
   hd(slot4<-value) -> layer_thickness;
  enddefmethod;

endflavour;

;;;    10. SUBSTRATE_PEAK_FRAME OBJECT

flavour SUBSTRATE_PEAK_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4;

 ;;; This method allows you to fill the SUBSTRATE_PEAK_FRAME
 defmethod fill_the_frame;
 lvars check quest1 quest2 quest3 quest4;

 "SUBSTRATE_PEAK_FRAME" -> name_of_frame;
 4 -> number_of_slots;

 ;;;    Screen display for the SUBSTRATE_PEAK_FRAME
 nl(3);
 pr('          DESCRIPTION OF THE SUBSTRATE PEAK  ');nl(3);


 ;;; SLOT1

 make_instance([QUESTION_SLOT])->slot1;
 '1. Please type in the Full Width at Half Maximum(FWHM) of the substrate
          peak (in arc seconds)' -> quest1;

 0 -> check;
 while (check=0)
 do
  slot1 <- fill_slot("FWHM",quest1);nl(2);
  test_number(slot1<-value) -> check;
 endwhile;

 ;;; SLOT2

 0 -> check;
 test_for_broadening(slot1<-value) -> check;

 if check=1
 then
  make_instance([SLOT])->slot2;
  slot2 <- fill_slot("PEAK_BROADENING",substrate_peak_broadening);
 else
  make_instance([Y_OR_N_QUESTION_SLOT])->slot2;

  '1a. Is the substrate peak broadened (Y/N)' -> quest2;
  slot2 <- fill_slot("PEAK_BROADENING",quest2);nl(2);
  if slot2<-value = [Y] or slot2<-value = [y]
  then
   1.0 -> slot2<-value;
  else
   0.0 -> slot2<-value
  endif;
 endif;


 ;;; SLOT3

 make_instance([QUESTION_SLOT])->slot3;
 '2. What is the integrated intensity of the substrate peak' -> quest3;
 0 -> check;
 while (check=0)
 do
  slot3 <- fill_slot("INTEGRATED_INTENSITY_OF_PEAK",quest3);nl(2);
  test_number(slot3<-value) -> check;
 endwhile;

 ;;; SLOT4

 make_instance([QUESTION_SLOT])->slot4;
 0 -> check;
 while (check=0)
 do
  pr('3. How ASYMMETRIC is the substrate peak?');nl(2);
  pr('    (a) Extremely asymmetric;');nl(1);
  pr('    (b) Very asymmetric;');nl(1);
  pr('    (c) Pretty asymmetric;');nl(1);
  pr('    (d) Just asymmetric; or');nl(1);
  pr('    (e) Symmetric.');nl(2);

  'Type the letter of your choice ...'->quest4;
  slot4 <- fill_slot("ASYMMETRY_IN_PEAK",quest4);nl(2);
  verify_input(slot4<-value) -> check
 endwhile;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  hd(slot1<-value) -> substrate_FWHM;
  slot2<-value -> substrate_peak_broadening;
  hd(slot3<-value) -> substrate_integrated_intensity_of_peak;
  assign_fuzzy_value(slot4<-value,fuzzy_asymmetry)
          -> substrate_asymmetry_in_peak;
 enddefmethod;

endflavour;


;;;    11. LAYER_PEAK_FRAME OBJECT

flavour LAYER_PEAK_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5 slot6;

 ;;; This method allows you to fill the LAYER_PEAK_FRAME
 defmethod fill_the_frame;
 lvars check quest1 quest3 quest4 quest5 quest6;

 "LAYER_PEAK_FRAME" -> name_of_frame;
 6 -> number_of_slots;


 ;;;    Screen display for the LAYER_PEAK_FRAME
 nl(3);
 pr('          DESCRIPTION OF LAYER PEAK  ');nl(3);


 ;;; SLOT1

 make_instance([QUESTION_SLOT])->slot1;
 '1. Please type in the Full Width at Half Maximum(FWHM) of the layer peak
          (in arc seconds)' -> quest1;
 0 -> check;
```

```
while (check=0)
do
  slot1 <- fill_slot("FWHM",quest1);nl(2);
  test_number(slot1<-value) -> check;
endwhile;

;;; SLOT2

make_instance([SLOT])->slot2;
slot2 <- fill_slot("PEAK_BROADENING",[N]);nl(2);

;;; SLOT3

make_instance([QUESTION_SLOT])->slot3;
'2. What is the integrated intensity of the layer peak' -> quest3;
0 -> check;
while (check=0)
do
  slot3 <- fill_slot("INTEGRATED_INTENSITY_OF_PEAK",quest3);nl(2);
  test_number(slot3<-value) -> check
endwhile;

;;; SLOT4

make_instance([QUESTION_SLOT])->slot4;
0 -> check;
while (check=0)
do
  pr('3. How ASYMMETRIC is the layer peak?');nl(2);
  pr('   (a) Extremely asymmetric;');nl(1);
  pr('   (b) Very asymmetric;');nl(1);
  pr('   (c) Pretty asymmetric;');nl(1);
  pr('   (d) Just asymmetric; or');nl(1);
  pr('   (e) Symmetric.');nl(2);

  'Type the letter of your choice ...'->quest4;
  slot4 <- fill_slot("ASYMMETRY_IN_PEAK",quest4);nl(2);
  verify_input(slot4<-value) -> check
endwhile;


;;; SLOT5

make_instance([QUESTION_SLOT])->slot5;
0 -> check;
while (check=0)
do
  pr('4. To what degree is the layer peak WEDGE SHAPED?');nl(2);
  pr('   (a) There is an extreme degree of wedging;');nl(1);
  pr('   (b) The peak is very obviously wedge shaped;');nl(1);
  pr('   (c) The peak is wedge shaped to a degree;');nl(1);
  pr('   (d) The peak could just about be described as wedge shaped; or');nl(1);
  pr('   (e) This peak is definitely not wedge shaped.');nl(2);

  'Type the letter of your choice ...'->quest5;
  slot5 <- fill_slot("WEDGE_SHAPED_PEAK",quest5);nl(2);
  verify_input(slot4<-value) -> check
endwhile;


;;; SLOT6

make_instance([Y_OR_N_QUESTION_SLOT])->slot6;
'5. Is the layer peak SPLIT (Y/N) ' -> quest6;
slot6 <- fill_slot("SPLIT_PEAK",quest6);nl(2);

enddefmethod;

;;; Transform the frame into a logic-based format
defmethod logic_format;
  hd(slot1<-value) -> layer_FWHM;
  assign_upper_case(slot2<-value) -> layer_peak_broadening;
  hd(slot3<-value) -> layer_integrated_intensity_of_peak;
  assign_fuzzy_value(slot4<-value,fuzzy_asymmetry)
        -> layer_asymmetry_in_peak;
  assign_fuzzy_value(slot5<-value,fuzzy_wedge_shaped)
        -> layer_wedge_shaped_peak;
  assign_upper_case(slot6<-value) -> layer_split_peak;
enddefmethod;

endflavour;
```

```
;;;    12. MULTIPLE_LAYER_PEAKS_FRAME OBJECT

flavour MULTIPLE_LAYER_PEAKS_FRAME isa FRAME;
  ivars slot1 slot2 slot3 slot4 slot5;

  ;;; This method allows you to fill the MULTIPLE_LAYER_PEAKS_FRAME
  defmethod fill the_frame;
  lvars check quest1 quest2 quest3 quest4 quest5;

  "MULTIPLE_LAYER_PEAKS_FRAME" -> name_of_frame;
  5 -> number_of_slots;

  ;;;    Screen display for the MULTIPLE_LAYER_PEAKS_FRAME
  nl(3);
  pr('          DESCRIPTION OF MULTIPLE LAYER PEAKS  ');nl(3);


  ;;; SLOT1

  make_instance([Y_OR_N_QUESTION_SLOT])->slot1;
  '1. Are any of the layer peaks broadened (Y/N)' -> quest1;
  slot1 <- fill_slot("PEAK_BROADENING",quest1);nl(2);

  ;;; SLOT2

  make_instance([Y_OR_N_QUESTION_SLOT])->slot2;
  '2. Are any of the layer peaks asymmetric (Y/N)' -> quest2;
  slot2 <- fill_slot("ANY_ASYMMETRY",quest2);nl(2);


  ;;; SLOT3

  if slot2<-value=[y] or slot2<-value=[Y]
  then
    make_instance([QUESTION_SLOT])->slot3;
    0 -> check;
    while (check=0)
    do
      pr(' How ASYMMETRIC is the layer peak?');nl(2);
      pr('   (a) Extremely asymmetric;');nl(1);
      pr('   (b) Very asymmetric;');nl(1);
      pr('   (c) Pretty asymmetric;');nl(1);
      pr('   (d) Just asymmetric; or');nl(1);
      pr('   (e) Symmetric.');nl(2);

      'Type the letter of your choice ...'->quest3;
      slot3 <- fill_slot("ASYMMETRY_IN_PEAK",quest3);nl(2);
      verify_input(slot3<-value) -> check
    endwhile;
  else
    make_instance([SLOT]) -> slot3;
    slot3 <- fill_slot("ASYMMETRY_IN_PEAK",[F]);
  endif;

  ;;; SLOT4

  make_instance([Y_OR_N_QUESTION_SLOT])->slot4;
  '3. Are any of the layer peaks wedge shaped (Y/N)' -> quest4;
  slot4 <- fill_slot("ANY_WEDGE_SHAPED_PEAKS",quest4);nl(2);

  ;;; SLOT5

  if slot4<-value=[y] or slot4<-value=[Y]
  then
    make_instance([QUESTION_SLOT])->slot5;
    0 -> check;
    while (check=0)
    do
      pr(' To what degree is the layer peak WEDGE SHAPED?');nl(2);
      pr('   (a) Extremely wedge shaped;');nl(1);
      pr('   (b) Very wedge shaped;');nl(1);
      pr('   (c) Pretty wedge shaped;');nl(1);
      pr('   (d) Just about wedge shaped; or');nl(1);
      pr('   (e) Not wedge shaped at all.');nl(2);

      'Type the letter of your choice ...'->quest5;
      slot5 <- fill_slot("WEDGE_SHAPED_PEAK",quest5);nl(2);
      verify_input(slot5<-value) -> check
    endwhile;
  else
    make_instance([SLOT]) -> slot5;
    slot5 <- fill_slot("WEDGE_SHAPED_PEAK",[F]);
  endif;
  enddefmethod;
```

```
  ;;; Transform the frame into a logic-based format
  defmethod logic_format;
   assign_upper_case(slot1 <-value) -> multiple_layers_peak_broadening;
   if multiple_layers_peak_broadening = [Y]
   then
    1 -> multiple_layers_peak_broadening
   else
    0 -> multiple_layers_peak_broadening
   endif;
   assign_upper_case(slot2 <-value) -> multiple_layers_any_asymmetry;
   if multiple_layers_any_asymmetry = [Y]
   then
    1 -> multiple_layers_any_asymmetry
   else
    0 -> multiple_layers_any_asymmetry
   endif;
   assign_fuzzy_value(slot3 <-value,fuzzy_asymmetry)
         -> multiple_layers_asymmetry_in_peak;
   assign_upper_case(slot4 <-value)
         -> multiple_layers_any_wedge_shaped_peaks;
   if multiple_layers_any_wedge_shaped_peaks = [Y]
   then
    1 -> multiple_layers_any_wedge_shaped_peaks
   else
    0 -> multiple_layers_any_wedge_shaped_peaks
   endif;
   assign_fuzzy_value(slot5 <-value,fuzzy_wedge_shaped)
         -> multiple_layers_wedge_shaped_peak;
  enddefmethod;

endflavour;


;;;    13. SINGLE_SATELLITE_PEAK_FRAME OBJECT

flavour SINGLE_SATELLITE_PEAK_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4;

 ;;; This method allows you to fill the SINGLE_SATELLITE_PEAK_FRAME
 defmethod fill_the_frame;
 lvars check quest1 quest2 quest3 quest4;

 "SINGLE_SATELLITE_PEAK_FRAME" -> name_of_frame;
 4 -> number_of_slots;

 make_instance([QUESTION_SLOT])->slot1;
 '    (a) Please type in the intensity of the satellite peak' -> quest1;
 0 -> check;
 while (check=0)
 do
  slot1 <- fill_slot("INTENSITY_OF_THE_PEAK",quest1);nl(2);
  test_number(slot1 <-value) -> check
 endwhile;

 make_instance([QUESTION_SLOT])->slot2;
 '    (b) What is the Full Width at Half Maximum (FWHM) of the satellite
            peak' -> quest2;
 0 -> check;
 while (check=0)
 do
  slot2 <- fill_slot("SATELLITE_FWHM",quest2);nl(2);
  test_number(slot2 <-value) -> check
 endwhile;

 make_instance([QUESTION_SLOT])->slot3;
 '    (c) What is the order of this satellite peak' -> quest3;
 0 -> check;
 while (check=0)
 do
  slot3 <- fill_slot("SATELLITE_ORDER",quest3);nl(2);
  test_integer(slot3 <-value) -> check
 endwhile;

 make_instance([QUESTION_SLOT])->slot4;
 '    (d) What is the position of this satellite peak (in arc secs)' -> quest4;
 0 -> check;
 while (check=0)
 do
  slot4 <- fill_slot("SATELLITE_POSITION",quest4);nl(2);
  test_number(slot4 <-value) -> check
 endwhile;

 enddefmethod;

  ;;; Transform the frame into a logic-based format
  defmethod logic_format;
   hd(slot1 <-value) -> satellite_intensity_of_peak;
   hd(slot2 <-value) -> satellite_FWHM;
   hd(slot3 <-value) -> satellite_order;
   hd(slot4 <-value) -> satellite_position;
  enddefmethod;

endflavour;


;;;    14. SATELLITE_PEAKS_FRAME OBJECT

flavour SATELLITE_PEAKS_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5 slot6 slot7 slot8 slot9 slot10;
 ivars slot11;

 ;;; This method allows you to fill the SATELLITE_PEAKS_FRAME
 defmethod fill_the_frame;
 lvars check quest1 quest2 quest3 quest4 quest9 quest10 quest11;
 lvars number_of_peaks height width area satellite_area_under_curve;
 lvars relative_height relative_width relative_area;
 lvars i single_satellite_peak_frame;

 "SATELLITE_PEAKS_FRAME" -> name_of_frame;
 11 -> number_of_slots;


 ;;;    Screen display for the SATELLITE_PEAKS_FRAME
 nl(3);
 pr('          DESCRIPTION OF SATELLITE PEAKS  ');nl(3);

 make_instance([QUESTION_SLOT])->slot1;
 0 -> check;
 while (check=0)
 do
  pr('1. Please describe the VISIBILITY of the satellite peaks?');nl(2);
  pr('    (a) Satellite peaks have very high intensity;');nl(1);
  pr('    (b) Satellite peaks are clearly visible;');nl(1);
  pr('    (c) Peaks are obvious, but not clear;');nl(1);
  pr('    (d) Peaks are very unclear; or');nl(1);
  pr('    (e) Cannot see any satellite peaks at all.');nl(2);

  'Type the letter of your choice ...'->quest1;
  slot1 <- fill_slot("VISIBILITY_OF_SATELLITE_PEAKS",quest1);nl(2);
  verify_input(slot1 <-value) -> check
 endwhile;


 if not(slot1 <-value=[e] or slot1 <-value=[E])
 then
  make_instance([QUESTION_SLOT])->slot2;
  '2. What is the spacing between satellite peaks (in arc secs) '->quest2;
  0 -> check;
  while (check=0)
  do
   slot2 <- fill_slot("SEPARATION_OF_SATELLITE_PEAKS",quest2);
   nl(2);
   test_number(slot2 <-value) -> check
  endwhile;
 else
  make_instance([SLOT]) -> slot2;
  slot2 <- fill_slot("SEPARATION_OF_SATELLITE_PEAKS",[0]);
 endif;

 if not(slot1 <-value=[e] or slot1 <-value=[E])
 then
  make_instance([QUESTION_SLOT])->slot3;
  0 -> check;
  while (check=0)
  do
   pr('3. How would you describe the relative intensities of the plus and minus
          satellites?');nl(2);
   pr('    (a) Extremely asymmetric;');nl(1);
   pr('    (b) Very asymmetric;');nl(1);
   pr('    (c) Pretty asymmetric;');nl(1);
   pr('    (d) Just asymmetric; or');nl(1);
   pr('    (e) Symmetric.');nl(2);

   'Type the letter of your choice ...'->quest3;
    slot3 <-
fill_slot("ASYMMETRY_OF_PLUS_AND_MINUS_SATELLITES",quest3);
   nl(2);
```

```
   verify_input(slot3<-value) -> check
  endwhile;
 else
  make_instance([SLOT]) -> slot3;
  slot3 <-
  fill_slot("ASYMMETRY_OF_PLUS_AND_MINUS_SATELLITES",[F]);
 endif;

 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  make_instance([QUESTION_SLOT])->slot4;
  '4. How many satellite peaks can you see (estimate)' -> quest4;
  0 -> check;
  while (check=0)
  do
   slot4 <- fill_slot("NUMBER_OF_SATELLITE_PEAKS",quest4);nl(2);
   test_integer(slot4<-value) -> check
  endwhile;
 else
  make_instance([SLOT]) -> slot4;
  slot4 <- fill_slot("NUMBER_OF_SATELLITE_PEAKS",[0]);
 endif;

 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  make_instance([SLOT])->slot5;
  slot5<-fill_slot("NUMBER_OF_SATELLITE_PEAKS_USED",[2]);nl(3);
 else
  make_instance([SLOT]) -> slot5;
  slot5 <- fill_slot("NUMBER_OF_SATELLITE_PEAKS_USED",[0]);
 endif;

 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  pr('5. DESCRIBE TWO SATELLITE PEAKS')
 endif;

 bd(slot5<-value) -> number_of_peaks;

 0 -> relative_height;
 0 -> relative_width;
 0 -> relative_area;

 0 -> height;
 0 -> width;
 0 -> area;

 1 -> i;
 until i > number_of_peaks
 do
  nl(3);pr('      SATELLITE PEAK NUMBER:  ');pr(i);nl(2);

  make_instance([SINGLE_SATELLITE_PEAK_FRAME])
        -> single_satellite_peak_frame;
  single_satellite_peak_frame <- fill_the_frame;
  single_satellite_peak_frame <- logic_format;

  ;;;*****
  ;;;CALCULATE RELATIVE MEASURES
  ;;;*****

  (satellite_intensity_of_peak * satellite_FWHM)/2
         -> satellite_area_under_curve;

  (height/satellite_intensity_of_peak) + relative_height -> relative_height;
  satellite_intensity_of_peak -> height;
  (width/satellite_FWHM) + relative_width -> relative_width;
  satellite_FWHM -> width;
  (area/satellite_area_under_curve) + relative_area -> relative_area;
  satellite_area_under_curve -> area;

  if i=1
  then
   satellite_order -> satellite_order_1;
   satellite_position -> satellite_position_1
  else
   satellite_order -> satellite_order_2;
   satellite_position -> satellite_position_2
  endif;

  i+1 -> i
 enduntil;
```

```
 ;;; To stop division by zero
 if number_of_peaks = 0 or number_of_peaks = 1
 then
  2 -> number_of_peaks
 endif;

 (relative_height/(number_of_peaks-1)) -> relative_height;
 (relative_width/(number_of_peaks-1)) -> relative_width;
 (relative_area/number_of_peaks-1) -> relative_area;

 make_instance([SLOT])->slot6;
 make_instance([SLOT])->slot7;
 make_instance([SLOT])->slot8;

 slot6 <-
 fill_slot("RELATIVE_INTENSITIES_OF_SATELLITES",relative_height);
 slot7<-fill_slot("RELATIVE_WIDTHS_OF_SATELLITES",relative_width);
 slot8 <-
 fill_slot("RELATIVE_INTEGRATED_INTENSITIES_OF_SATELLITES",
 relative_area);
 nl(2);


 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  make_instance([Y_OR_N_QUESTION_SLOT])->slot9;
  pr('6. Are there any SUBSIDIARY INTERFERENCE EFFECTS on the
          satellites');nl(1);
  '   (Y/N)' -> quest9;
  slot9 <-
  fill_slot("SUBSIDIARY_INTERFERENCE_EFFECTS",quest9);
  nl(2);
 else
  make_instance([SLOT]) -> slot9;
  slot9 <- fill_slot("SUBSIDIARY_INTERFERENCE_EFFECTS",[N]);
 endif;

 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  make_instance([Y_OR_N_QUESTION_SLOT])->slot10;
  '7. Do you detect any PEAK SPLITTING of satellites (Y/N) ' -> quest10;
  slot10<-fill_slot("PEAK_SPLITTING_OF_SATELLITES",quest10);nl(2);
 else
  make_instance([SLOT]) -> slot10;
  slot10 <- fill_slot("PEAK_SPLITTING_OF_SATELLITES",[N]);
 endif;

 if not(slot1<-value=[e] or slot1<-value=[E])
 then
  make_instance([Y_OR_N_QUESTION_SLOT])->slot11;
  '8. Are the higher ordered satellites BROADENED (Y/N) ' -> quest11;
  slot11 <-
  fill_slot("BROADENING_OF_HIGHER_ORDERED_SATELLITES",
  quest11);
 nl(2);
 else
  make_instance([SLOT]) -> slot11;
  slot11 <-
  fill_slot("BROADENING_OF_HIGHER_ORDERED_SATELLITES",[N]);
 endif;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  assign_fuzzy_value(slot1<-value,fuzzy_visibility) -> satellite_visibility;
  bd(slot2<-value) -> satellite_spacing_of_peaks;
  if satellite_spacing_of_peaks > 1.5
  then
   abe(1.0-(1.0/satellite_spacing_of_peaks)) -> separation_of_satellite_peaks
  elseif satellite_spacing_of_peaks > 0.5
  then
   abe(satellite_spacing_of_peaks/5.0) -> separation_of_satellite_peaks
  else
   0.01 -> separation_of_satellite_peaks
  endif;
  assign_fuzzy_value(slot3<-value,fuzzy_plus_minus_asymmetry) ->
          satellite_asymmetry_of_plus_and_minus_peaks;
  bd(slot4<-value) -> satellite_number_of_peaks;
  slot6<-value -> satellite_relative_intensities;
  slot7<-value -> satellite_relative_width_of_peaks;
  slot8<-value -> satellite_relative_integrated_intensities;
  assign_upper_case(slot9<-value)->satellite_subsidiary_interference_effects;
  if satellite_subsidiary_interference_effects = [Y]
```

```
    then
      1 -> satellite_subsidiary_interference_effects
    else
      0 -> satellite_subsidiary_interference_effects
    endif;
    assign_upper_case(slot10<-value) -> satellite_peak_splitting;
    if satellite_peak_splitting = [Y]
    then
      1 -> satellite_peak_splitting
    else
      0 -> satellite_peak_splitting
    endif;
    assign_upper_case(slot11<-value)
          -> satellite_broadening_of_higher_order_peaks;
    if satellite_broadening_of_higher_order_peaks = [Y]
    then
      1 -> satellite_broadening_of_higher_order_peaks
    else
      0 -> satellite_broadening_of_higher_order_peaks
    endif;
  enddefmethod;

endflavour;



;;;     15. ZERO_ORDER_PEAK_FRAME OBJECT

flavour ZERO_ORDER_PEAK_FRAME isa FRAME;
  ivars slot1 slot2 slot3 slot4;

  ;;; This method allows you to fill the ZERO_ORDER_PEAK_FRAME
  defmethod fill_the_frame;
  lvars check quest1 quest3 quest4;

  "ZERO_ORDER_PEAK_FRAME" -> name_of_frame;
  4 -> number_of_slots;

  ;;;     Screen display for the ZERO_ORDER_PEAK_FRAME
  nl(3);
  pr('          DESCRIPTION OF THE ZERO_ORDER PEAK  ');nl(3);


  make_instance([QUESTION_SLOT])->slot1;
  '1. Please type in the Full Width at Half Maximum (FWHM) of the zero_order
            peak (in arc seconds)' -> quest1;
  0 -> check;
  while (check=0)
  do
    slot1 <- fill_slot("FWHM",quest1);nl(2);
    test_number(slot1<-value) -> check
  endwhile;

  make_instance([SLOT])->slot2;
  slot2 <- fill_slot("PEAK_BROADENING",[N]);nl(2);

  make_instance([QUESTION_SLOT])->slot3;
  '3. What is the integrated intensity of the zero_order peak' -> quest3;
  0 -> check;
  while (check=0)
  do
    slot3 <- fill_slot("INTEGRATED_INTENSITY_OF_PEAK",quest3);nl(2);
    test_number(slot3<-value) -> check
  endwhile;

  make_instance([QUESTION_SLOT])->slot4;
  0 -> check;
  while (check=0)
  do
    pr('4. How ASYMMETRIC is the zero_order peak?');nl(2);
    pr('    (a) Extremely asymmetric;');nl(1);
    pr('    (b) Very asymmetric;');nl(1);
    pr('    (c) Pretty asymmetric;');nl(1);
    pr('    (d) Just asymmetric; or');nl(1);
    pr('    (e) Symmetric.');nl(2);

    'Type the letter of your choice ...'->quest4;
    slot4 <- fill_slot("ASYMMETRY_IN_PEAK",quest4);nl(2);
    verify_input(slot4<-value) -> check
  endwhile;

  enddefmethod;

  ;;; Transform the frame into a logic-based format
  defmethod logic_format;
    hd(slot1<-value) -> zero_order_FWHM;
    assign_upper_case(slot2<-value) -> zero_order_peak_broadening;
    hd(slot3<-value) -> zero_order_integrated_intensity_of_peak;
    assign_fuzzy_value(slot4<-value,fuzzy_asymmetry)
          -> zero_order_asymmetry_in_peak;
  enddefmethod;

endflavour;


;;;  16.  INTERFERENCE_FRINGES_FRAME OBJECT

flavour INTERFERENCE_FRINGES_FRAME isa FRAME;
  ivars slot1 slot2 slot3;

  ;;; This method allows you to fill the INTERFERENCE_FRINGES_FRAME
  defmethod fill_the_frame;
  lvars check quest1 quest2 quest3;

  "INTERFERENCE_FRINGES_FRAME" -> name_of_frame;
  3 -> number_of_slots;

  ;;;     Screen display for the INTERFERENCE_FRINGES_FRAME
  nl(3);
  pr('          DESCRIPTION OF THE INTERFERENCE FRINGES  ');nl(3);


  ;;; SLOT1

  make_instance([QUESTION_SLOT])->slot1;
  0 -> check;
  while (check=0)
  do
    pr('1. Are there INTERFERENCE FRINGES on the rocking curve?');nl(2);
    pr('    (a) Very large number of fringes;');nl(1);
    pr('    (b) A significant number of fringes;');nl(1);
    pr('    (c) Some fringes;');nl(1);
    pr('    (d) There MAY BE fringes present');nl(1);
    pr('        BUT I am not quite certain; or');nl(1);
    pr('    (e) No fringes at all.');nl(2);

    'Type the letter of your choice ...'->quest1;
    slot1 <- fill_slot("INTERFERENCE_FRINGES",quest1);nl(2);
    verify_input(slot1<-value) -> check
  endwhile;

  ;;; SLOT2

  if not(slot1<-value=[e] or slot1<-value=[E])
  then
    make_instance([QUESTION_SLOT])->slot2;
    '2. What is the spacing of the interference fringes (in arc secs) '->quest2;
    0 -> check;
    while (check=0)
    do
      slot2 <- fill_slot("SPACING_OF_INTERFERENCE_FRINGES",quest2);
      nl(2);
      test_number(slot2<-value) -> check
    endwhile;
  else
    make_instance([SLOT]) -> slot2;
    slot2 <- fill_slot("SPACING_OF_INTERFERENCE_FRINGES",[0]);
  endif;


  ;;; SLOT3

  if not(slot1<-value=[e] or slot1<-value=[E])
  then
    make_instance([QUESTION_SLOT])->slot3;
    0 -> check;
    while (check=0)
    do
      pr('3. Please describe the VISIBILITY of interference fringes on the layer
            peak?');nl(2);
      pr('    (a) Fringes have very high intensity;');nl(1);
      pr('    (b) Fringes are clearly visible;');nl(1);
      pr('    (c) Fringes are obvious, but not clear;');nl(1);
      pr('    (d) Fringes are very unclear; or');nl(1);
      pr('    (e) Cannot see any fringes at all.');nl(2);

      'Type the letter of your choice ...'->quest3;
```

```
    slot3<-fill_slot("VISIBILITY_OF_INTERFERENCE_FRINGES",quest3);
    nl(2);
    verify_input(slot3<-value) -> check
  endwhile;
 else
  make_instance([SLOT]) -> slot3;
  slot3 <- fill_slot("VISIBILITY_OF_INTERFERENCE_FRINGES",[F]);
 endif;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  assign_fuzzy_value(slot1<-value,fuzzy_interference)->interference_fringes;
  hd(slot2<-value) -> spacing_of_interference_fringes;
  assign_fuzzy_value(slot3<-value,fuzzy_visibility)
       -> visibility_of_interference_fringes;
 enddefmethod;

endflavour;

;;; ____________________________________________

;;;    17. SUBSTRATE_ONLY_FRAME OBJECT

flavour SUBSTRATE_ONLY_FRAME isa FRAME;
 ivars slot1 slot2;

 ;;; This method allows you to fill the SUBSTRATE_ONLY_FRAME
 defmethod fill_the_frame;
 lvars experiment_frame substrate_peak_frame interference_fringes_frame;
 lvars check quest1 name_of_slot2;

 "SUBSTRATE_ONLY_FRAME" -> name_of_frame;
 2 -> number_of_slots;

 if x=1
 then
  ;;; FILL THE EXPERIMENT_FRAME: THIS FILLS SLOTS 1 - 12
  make_instance([EXPERIMENT_FRAME]) -> experiment_frame;
  experiment_frame <- fill_the_frame;
  experiment_frame <- logic_format;
 endif;

 0 -> check;
 while (check=0)
 do

  ;;;    Screen display for the SUBSTRATE_ONLY_FRAME
  nl(3);
  pr('          SUBSTRATE ONLY STRUCTURE');nl(3);


  ;;; FIRST QUESTION: FILLS SLOTS 1 & 2

  pr('1. How many peaks are there in the rocking curve?');nl(2);
  pr('   (a) one peak;');nl(1);
  pr('   (b) more than one peak;');nl(1);
  pr('   (c) no peaks whatsoever.');nl(2);

  ;;;    Assign leading question to a variable: quest1
  'Type the letter of your choice ...'->quest1;

  make_instance([QUESTION_SLOT])->slot1;
  slot1 <- fill_slot("NUMBER_OF_PEAKS",quest1);nl(2);

  make_instance([SLOT])->slot2;
  "HOW_MANY_PEAKS"->name_of_slot2;

  1 -> check;
  if (slot1<-value=[a] or slot1<-value=[A])
  then
   slot2 <- fill_slot(name_of_slot2,'one peak');
   1 -> number_of_peaks_is_one;
  elseif (slot1<-value=[b] or slot1<-value=[B])
  then
   slot2 <- fill_slot(name_of_slot2,'more than one peak');
   1 -> number_of_peaks_is_more_than_one;
  elseif (slot1<-value=[c] or slot1<-value=[C])
  then
   slot2 <- fill_slot(name_of_slot2,'no peaks whatsoever');
   1 -> number_of_peaks_is_none;
   return()
  else
   nl(4);pr('You have typed an incorrect answer to this question ...');
   nl(2);pr('Please, try again ...');nl(1);
   0 -> check;
  endif;
 endwhile;

 if (slot2<-value = 'one peak' or slot2<-value = 'more than one peak')
 then
  make_instance([SUBSTRATE_PEAK_FRAME]) -> substrate_peak_frame;
  substrate_peak_frame <- fill_the_frame;
  substrate_peak_frame <- logic_format;
  cancel substrate_peak_frame;

  make_instance([INTERFERENCE_FRINGES_FRAME])
       -> interference_fringes_frame;
  interference_fringes_frame <- fill_the_frame;
  interference_fringes_frame <- logic_format;
 endif;

 enddefmethod;


 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  0 -> number_of_layers;
 enddefmethod;

endflavour;


;;;    18. SINGLE_LAYER_FRAME OBJECT

flavour SINGLE_LAYER_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5;

 ;;; This method allows you to fill the SINGLE_LAYER_FRAME
 defmethod fill_the_frame;
 lvars experiment_frame;
 lvars substrate_peak_frame layer_peak_frame interference_fringes_frame;

 lvars i check quest1 name_of_slot2 quest3 quest4 quest5;

 "SINGLE_LAYER_FRAME" -> name_of_frame;
 34 -> number_of_slots;


 ;;;    Screen display for the SINGLE_LAYER_FRAME
 nl(3);
 pr('          A SINGLE LAYER STRUCTURE');nl(3);

 if x=1
 then
  make_instance([EXPERIMENT_FRAME]) -> experiment_frame;
  experiment_frame <- fill_the_frame;
  experiment_frame <- logic_format;
 endif;

 if x=1
 then
  1 -> i;
  make_instance([LAYER_FRAME]) -> layer_frame;
  layer_frame <- fill_the_frame;
  layer_frame <- logic_format;
 endif;

 0 -> check;
 while (check=0)
 do
  nl(3);
  pr('          DESCRIPTION OF THE CURVE');nl(3);

  pr('1. How many peaks are there in the rocking curve?');nl(2);
  pr('   (a) one peak;');nl(1);
  pr('   (b) two peaks;');nl(1);
  pr('   (c) more than two peaks;');nl(1);
  pr('   (d) no peaks whatsoever.');nl(2);

  'Type the letter of your choice ...'->quest1;
  make_instance([QUESTION_SLOT])->slot1;
  slot1 <- fill_slot("NUMBER_OF_PEAKS",quest1);nl(2);

  make_instance([SLOT])->slot2;
  "HOW_MANY_PEAKS"->name_of_slot2;
```

```
 1 -> check;
 if (slot1 <-value=[a] or slot1 <-value=[A])
 then
  slot2 <- fill_slot(name_of_slot2,'one peak');
  1 -> number_of_peaks_is_one;
 elseif (slot1 <-value=[b] or slot1 <-value=[B])
 then
  slot2 <- fill_slot(name_of_slot2,'two peaks');
  1 -> number_of_peaks_is_two;
 elseif (slot1 <-value=[c] or slot1 <-value=[C])
 then
  slot2 <- fill_slot(name_of_slot2,'more than two peaks');
  1 -> number_of_peaks_is_more_than_two;
 elseif (slot1 <-value=[d] or slot1 <-value=[D])
 then
  slot2 <- fill_slot(name_of_slot2,'no peaks whatsoever');
  1 -> number_of_peaks_is_none;
 else
  nl(4);pr('You have typed an incorrect answer to this question ...');
  nl(2);pr('Please, try again ...');nl(2);
  0 -> check
 endif;
endwhile;


if (slot2 <-value = 'one peak' or slot2 <-value = 'two peaks' or slot2 <-value
          = 'more than two peaks')
then
 make_instance([Y_OR_N_QUESTION_SLOT])->slot3;
 '2. Does the layer peak overlap with the substrate peak (Y/N)' -> quest3;
 slot3 <- fill_slot("OVERLAP",quest3);nl(3);

 if slot2 <-value = 'one peak'
 then
  make_instance([SLOT])->slot4;
  slot4 <-fill_slot("PEAK_SPLITTING",[0])
 else
  make_instance([QUESTION_SLOT])->slot4;
  '2(a). Type in a measure of PEAK SEPARATION between the substrate and
         layer peaks (in arc seconds):' -> quest4;
  0 -> check;
  while (check=0)
  do
   slot4 <- fill_slot("PEAK_SPLITTING",quest4);nl(3);
   test_number(slot4 <-value) -> check;
  endwhile;
 endif;

 if x=1
 then
  make_instance([Y_OR_N_QUESTION_SLOT])->slot5;
  '3. Is there a possibility of a RELAXED layer at the interface (Y/N)?'
          -> quest5;
  slot5 <- fill_slot("RELAXED_LAYER",quest5);

  if slot5 <-value = 'Y'
  then
   1 -> relaxed_mismatch_is_high;
  else
   0 -> relaxed_mismatch_is_high;
  endif;
 else
  make_instance([SLOT])->slot5;
  slot5 <- fill_slot("RELAXED_LAYER",relaxed_layer)
 endif;

 make_instance([SUBSTRATE_PEAK_FRAME]) -> substrate_peak_frame;
 substrate_peak_frame <- fill_the_frame;
 substrate_peak_frame <- logic_format;
 cancel substrate_peak_frame;

 if (slot2 <-value = 'two peaks' or slot2 <-value = 'more than two peaks')
 then
  make_instance([LAYER_PEAK_FRAME]) -> layer_peak_frame;
  layer_peak_frame <- fill_the_frame;
  layer_peak_frame <- logic_format;
 endif;

 make_instance([INTERFERENCE_FRINGES_FRAME])
     -> interference_fringes_frame;
 interference_fringes_frame <- fill_the_frame;
 interference_fringes_frame <- logic_format;
 cancel interference_fringes_frame;
```

```
 if spacing_of_interference_fringes < 3
 then
  1 -> spacing_of_interference_fringes_is_low;
 else
  0 -> spacing_of_interference_fringes_is_low;
 endif;
else
  make_instance([SLOT])->slot4;
  slot4 <- fill_slot("PEAK_SPLITTING",[0]);
  make_instance([SLOT])->slot5;
  slot5 <- fill_slot("RELAXED_LAYER",[relaxed_layer])
endif;

enddefmethod;


;;; Transform the frame into a logic-based format
defmethod logic_format;
 lvars a b;

 1 -> number_of_layers;

 if substrate_material = layer_material
 then
  1 -> substrate_material_equal_to_layer;
 else
  0 -> substrate_material_equal_to_layer;
 endif;

 if layer_thickness > 0.5
 then
  1 -> layer_thickness_greater_than_half_micron;
 else
  0 -> layer_thickness_greater_than_half_micron;
 endif;

 if layer_thickness < 5.0
 then
  1 -> layer_thickness_less_than_5_microns;
 else
  0 -> layer_thickness_less_than_5_microns;
 endif;

 slot2 <-value -> number_of_peaks;
 assign_upper_case(slot3 <-value) -> substrate_layer_peak_overlap;
 hd(slot4 <-value) -> peak_splitting;

 if peak_splitting < 150
 then
  1 -> peak_separation_is_low;
 else
  0 -> peak_separation_is_low;
 endif;

 if peak_splitting > 150 or peak_splitting = 150
 then
  1 -> peak_splitting_is_high;
 else
  0 -> peak_splitting_is_high;
 endif;

 if peak_splitting = 0
 then
  1 -> peak_splitting_is_zero;
 else
  0 -> peak_splitting_is_zero;
 endif;

 assign_upper_case(slot5 <-value) -> relaxed_layer;

 3 * layer_FWHM -> a;
 3 * substrate_FWHM -> b;
 if a > b
 then
  a -> three_times_width_of_peak;
 else
  b -> three_times_width_of_peak;
 endif;

 if peak_splitting < three_times_width_of_peak
 then
  1 -> peak_splitting_less_than_three_times_width_of_peak;
 else
  0 -> peak_splitting_less_than_three_times_width_of_peak;
```

```
  endif;

  if layer_integrated_intensity_of_peak = 0
  then
   1 -> layer_integrated_intensity_of_peak_is_zero;
  else
   0 -> layer_integrated_intensity_of_peak_is_zero;
  endif;

  if number_of_peaks = 'one peak'
  then
   0.0 -> intensity_of_layer_peak
  else
   if ((2 * layer_integrated_intensity_of_peak) / (layer_FWHM * 100)) > 100
   then
    0.9 -> intensity_of_layer_peak;
   elseif ((2 * layer_integrated_intensity_of_peak) / (layer_FWHM * 100))
            > 60
   then
    0.7 -> intensity_of_layer_peak;
   elseif ((2 * layer_integrated_intensity_of_peak) / (layer_FWHM * 100))
            > 40
   then
    0.5 -> intensity_of_layer_peak;
   elseif ((2 * layer_integrated_intensity_of_peak) / (layer_FWHM * 100))
            > 20
   then
    0.3 -> intensity_of_layer_peak;
   else
    0.1 -> intensity_of_layer_peak;
   endif;
  endif;

 enddefmethod;

endflavour;


;;;    19. A_NUMBER_OF_LAYERS_FRAME OBJECT

flavour A_NUMBER_OF_LAYERS_FRAME isa FRAME;
 ivars slot1 slot2;

 ;;; This method allows you to fill the A_NUMBER_OF_LAYERS_FRAME
 defmethod fill_the_frame;
  lvars check i quest1;

 "A_NUMBER_OF_LAYERS_FRAME" -> name_of_frame;
 2 -> number_of_slots;


 make_instance([QUESTION_SLOT])->slot1;
 '2. How many layers are there in the structure '->quest1;
 0 -> check;
 while (check=0)
 do
  slot1 <- fill_slot("NUMBER_OF_LAYERS",quest1);nl(2);
  test_integer(slot1 <-value) -> check
 endwhile;

 ;;; FILL LAYER_FRAMEs:

 hd(slot1 <-value) -> number_of_layers;
 newarray([1 ^number_of_layers]) -> layer_frame;
 1 -> i;
 until i > number_of_layers
 do
  nl(2);
  pr('_____');
  nl(2);
  pr('3. Layer Number: ');pr(i);nl(2);

  make_instance([LAYER_FRAME]) -> layer_frame(i);
  layer_frame(i) <- fill_the_frame;
  i+1 -> i
 enduntil;

 ;;; SLOT 2

 make_instance([SLOT])->slot2;
 slot2 <- fill_slot("NUMBER_OF_BLOCKS",[1]);nl(2);

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  hd(slot1 <-value) -> number_of_layers;
  hd(slot2 <-value) -> number_of_blocks;
 enddefmethod;
endflavour;

;;;    20. MULTIPLE_LAYERS_SUB_FRAME OBJECT

flavour MULTIPLE_LAYERS_SUB_FRAME isa FRAME;
 ivars slot1 slot2;

 ;;; This method allows you to fill the MULTIPLE_LAYERS_SUB_FRAME
 defmethod fill_the_frame;
 lvars i experiment_frame a_number_of_layers_frame quest1 quest2 check;
 lvars substrate_peak_frame multiple_layer_peaks_frame;
 lvars interference_fringes_frame;

 "MULTIPLE_LAYERS_SUB_FRAME" -> name_of_frame;
 1 -> number_of_slots;

 if x=1
 then
  make_instance([EXPERIMENT_FRAME]) -> experiment_frame;
  experiment_frame <- fill_the_frame;
  experiment_frame <- logic_format
 endif;

 nl(3);
 pr('                    DESCRIPTION OF THE LAYERS IN THIS
STRUCTURE');nl(3);

 if x=1
 then
   make_instance([A_NUMBER_OF_LAYERS_FRAME])   ->
          a_number_of_layers_frame;
  a_number_of_layers_frame <- fill_the_frame;
  a_number_of_layers_frame <- logic_format
 endif;

 make_instance([QUESTION_SLOT]) -> slot1;
 '4. How many peaks in the curve' -> quest1;
 0 -> check;
 while (check=0)
 do
    slot1 <-fill_slot("NUMBER_OF_PEAKS",quest1);
  test_number(slot1 <-value) -> check
 endwhile;

 if hd(slot1 <-value) = 0
 then
  1 -> number_of_peaks_is_none;
 else
  0 -> number_of_peaks_is_none;
 endif;

 if hd(slot1 <-value) = (number_of_layers + 1)
 then
  1 -> correspondence_between_layers_and_peaks;
 else
  0 -> correspondence_between_layers_and_peaks;
 endif;

 make_instance([SUBSTRATE_PEAK_FRAME]) -> substrate_peak_frame;
 substrate_peak_frame <- fill_the_frame;
 substrate_peak_frame <- logic_format;

 make_instance([Y_OR_N_QUESTION_SLOT])->slot2;
 '4. Is the substrate peak split(Y/N)' -> quest2;
 slot2 <- fill_slot("SPLIT_SUBSTRATE_PEAK",quest2);
 if slot2<-value = 'Y'
 then
  1 -> split_substrate_peak
 else
  0 -> split_substrate_peak
 endif;

 make_instance([MULTIPLE_LAYER_PEAKS_FRAME])
       -> multiple_layer_peaks_frame;
 multiple_layer_peaks_frame <- fill_the_frame;
 multiple_layer_peaks_frame <- logic_format;

 make_instance([INTERFERENCE_FRINGES_FRAME])
       -> interference_fringes_frame;
```

```
  interference_fringes_frame <- fill_the_frame;
  interference_fringes_frame <- logic_format;

  enddefmethod;

endflavour;


;;;    21. BLOCK_OF_LAYERS_FRAME OBJECT

flavour BLOCK_OF_LAYERS_FRAME isa FRAME;
 ivars slot1 slot2;

 ;;; This method allows you to fill the BLOCK_OF_LAYERS_FRAME
 defmethod fill_the_frame;
 lvars i check quest1 quest2;

 "BLOCK_OF_LAYERS_FRAME" -> name_of_frame;
 2 -> number_of_slots;

 if number_of_layers=2
 then
  pr('1. There are TWO layers per block in an MQW structure.');nl(2);
  pr('   PLEASE DESCRIBE EACH LAYER');
  make_instance([SLOT])->slot1;
  slot1 <- fill_slot("NUMBER_OF_LAYERS",[2]);nl(2);
 else
  '1. How many layers in each block' -> quest1;
  make_instance([QUESTION_SLOT])->slot1;
  0 -> check;
  while (check=0)
  do
   slot1 <- fill_slot("NUMBER_OF_LAYERS",quest1);nl(2);
   test_integer(slot1<-value) -> check
  endwhile;
 endif;

 ;;; FILL LAYER_FRAMEs:

 hd(slot1<-value) -> number_of_layers;
 newarray([1 'number_of_layers]) -> layer_frame;

 1 -> i;
 until i > number_of_layers
 do
  nl(2);
  pr('2. Layer Number: ');pr(i);nl(2);
  make_instance([LAYER_FRAME]) -> layer_frame(i);
  layer_frame(i) <- fill_the_frame;
  i+1 -> i
 enduntil;
 nl(3);

 make_instance([QUESTION_SLOT])->slot2;
 '3. How many blocks in this structure' -> quest2;
 0 -> check;
 while (check=0)
 do
  slot2 <- fill_slot("NUMBER_OF_BLOCKS",quest2);nl(2);
  test_integer(slot2<-value) -> check
 endwhile;

 enddefmethod;


 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  hd(slot1<-value) -> number_of_layers;
  hd(slot2<-value) -> number_of_blocks;
 enddefmethod;

endflavour;


;;;    22. SUPERLATTICE_FRAME OBJECT

flavour SUPERLATTICE_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4;

 ;;; This method allows you to fill the SUPERLATTICE_FRAME
 defmethod fill_the_frame;
 lvars experiment_frame block_of_layers_frame;
 lvars satellite_peaks_frame substrate_peak_frame zero_order_peak_frame;
 lvars interference_fringes_frame;
 lvars check quest1 quest2 quest3 quest4;

 "SUPERLATTICE_FRAME" -> name_of_frame;
 3 -> number_of_slots;

 if x=1
 then
  make_instance([EXPERIMENT_FRAME]) -> experiment_frame;
  experiment_frame <- fill_the_frame;
  experiment_frame <- logic_format
 endif;

 nl(4);pr('DESCRIPTION OF BLOCK OF LAYERS');nl(3);

 if x=1
 then
  make_instance([BLOCK_OF_LAYERS_FRAME])
       -> block_of_layers_frame;
  block_of_layers_frame <- fill_the_frame;
  block_of_layers_frame <- logic_format
 endif;

 if x=1
 then
  make_instance([QUESTION_SLOT])->slot1;
  '4. Can you estimate the THICKNESS of the superlattice in microns'
            -> quest1;
  0 -> check;
  while (check=0)
  do
   slot1 <- fill_slot("THICKNESS_OF_SUPERLATTICE",quest1);nl(2);
   test_number(slot1<-value) -> check
  endwhile;

  if hd(slot1<-value) < 0.5
  then
   1 -> thickness_of_superlattice_less_than_half_micron;
  else
   0 -> thickness_of_superlattice_less_than_half_micron;
  endif;
 else
  make_instance([SLOT])->slot1;
  slot1 <-
  fill_slot("THICKNESS_OF_SUPERLATTICE",[^thickness_of_superlattice])
 endif;

 nl(3);
 pr('            GENERAL DESCRIPTION OF THE CURVE');nl(2);

 make_instance([Y_OR_N_QUESTION_SLOT])->slot2;
 '1. Are there MORE THAN TWO main peaks for the superlattice(Y/N) '
         -> quest2;
 slot2 <- fill_slot("MORE_THAN_TWO_PEAKS",quest2);nl(2);

 if slot2<-value = [N]
 then
  1 -> number_of_peaks_is_none;
 else
  0 -> number_of_peaks_is_none;
 endif;

 make_instance([Y_OR_N_QUESTION_SLOT])->slot3;
 '2. Can you identify the substrate peak, zero-order peak and the satellite peaks
           (Y/N) ' -> quest3;
 slot3 <- fill_slot("IDENTIFY_THE_PEAKS",quest3);nl(3);

 make_instance([SATELLITE_PEAKS_FRAME]) -> satellite_peaks_frame;
 satellite_peaks_frame <- fill_the_frame;
 satellite_peaks_frame <- logic_format;

 make_instance([SUBSTRATE_PEAK_FRAME]) -> substrate_peak_frame;
 substrate_peak_frame <- fill_the_frame;
 substrate_peak_frame <- logic_format;

 make_instance([ZERO_ORDER_PEAK_FRAME])->zero_order_peak_frame;
 zero_order_peak_frame <- fill_the_frame;
 zero_order_peak_frame <- logic_format;

 make_instance([QUESTION_SLOT]) -> slot4;
 nl(3);
 pr('            PEAK SPLITTING');nl(2);
 '1. Type in a measure of PEAK SEPARATION between the substrate and
```

```
  zero-order peaks(in arcs secs)' -> quest4;
 0 -> check;
 while (check = 0)
 do
  slot4 <- fill_slot("PEAK_SPLITTING",quest4);nl(2);
  test_number(slot4<-value) -> check
 endwhile;
 hd(slot4<-value) -> peak_splitting;
 ;;; Force the system to calculate mismatch ...
 0.2 -> experimental_mismatch;

 if type_of_structure_has_additional_layers = 1
 then
  make_instance([INTERFERENCE_FRINGES_FRAME])
       -> interference_fringes_frame;
  interference_fringes_frame <- fill_the_frame;
  interference_fringes_frame <- logic_format;
 endif;

 enddefmethod;


 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  hd(slot1<-value) -> thickness_of_superlattice;
  assign_upper_case(slot2<-value) -> more_than_two_peaks;
  if more_than_two_peaks = [Y]
  then
   1 -> more_than_two_peaks
  else
   0 -> more_than_two_peaks
  endif;
  assign_upper_case(slot3<-value) -> identify_the_peaks;
  if identify_the_peaks = [Y]
  then
   1 -> identify_the_peaks
  else
   0 -> identify_the_peaks
  endif;

  if satellite_spacing_of_peaks > 0
  then
   1 -> satellite_spacing_greater_than_zero;
  else
   0 -> satellite_spacing_greater_than_zero;
  endif;

  if satellite_broadening_of_higher_order_peaks = [Y]
  then
   1 -> satellite_broadening_of_higher_order_peaks;
  else
   0 -> satellite_broadening_of_higher_order_peaks;
  endif;

  if satellite_subsidiary_interference_effects = [Y]
  then
   1 -> satellite_subsidiary_interference_effects;
  else
   0 -> satellite_subsidiary_interference_effects;
  endif;

  if satellite_relative_width_of_peaks > 0
  then
   1 -> satellite_relative_width_of_peaks_greater_than_zero;
  else
   0 -> satellite_relative_width_of_peaks_greater_than_zero;
  endif;

  if satellite_relative_integrated_intensities > 0
  then
   1 -> satellite_relative_integrated_intensities_greater_than_zero;
  else
   0 -> satellite_relative_integrated_intensities_greater_than_zero;
  endif;

 enddefmethod;

endflavour;
```

```
;;;    23. MULTIPLE_LAYERS_FRAME OBJECT

flavour MULTIPLE_LAYERS_FRAME isa FRAME;
 ivars slot1;

 ;;; This method allows you to fill the MULTIPLE_LAYERS_FRAME
 defmethod fill_the_frame;
 lvars multiple_layers_sub_frame superlattice_frame;
 lvars quest1;

 "MULTIPLE_LAYERS_FRAME" -> name_of_frame;
 1 -> number_of_slots;

 if x=1
 then
  nl(3);
  pr('          DESCRIPTION OF THE LAYERS IN THIS
          STRUCTURE');nl(3);

  '1. Are the layers built up in blocks e.g. ABA etc.(Y/N)' -> quest1;
  make_instance([Y_OR_N_QUESTION_SLOT])->slot1;
  slot1 <- fill_slot("LAYERS_IN_BLOCKS",quest1);nl(2)
 else
  make_instance([SLOT])-> slot1;
  slot1 <- fill_slot("LAYERS_IN_BLOCKS",layers_in_blocks)
 endif;

 if slot1<-value=[n] or slot1<-value=[N]
 then
  make_instance([MULTIPLE_LAYERS_SUB_FRAME])
       -> multiple_layers_sub_frame;
  multiple_layers_sub_frame <- fill_the_frame;
 else
  1 -> type_of_structure_is_MQW;
  0 -> type_of_structure_is_multiple_layers;
  'MQW structure' -> type_of_structure;
  make_instance([SUPERLATTICE_FRAME]) -> superlattice_frame;
  superlattice_frame <- fill_the_frame;
  superlattice_frame <- logic_format;
 endif;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  assign_upper_case(slot1<-value) -> layers_in_blocks;
 enddefmethod;

endflavour;


;;;    24. MQW_STRUCTURE_FRAME OBJECT

flavour MQW_STRUCTURE_FRAME isa FRAME;

 ;;; This method allows you to fill the MQW_STRUCTURE_FRAME
 defmethod fill_the_frame;
 lvars superlattice_frame;

 "MQW_STRUCTURE_FRAME" -> name_of_frame;
 0 -> number_of_slots;

 if x = 1
 then
  ;;;    Screen display for the MQW_STRUCTURE_FRAME
  nl(3);
  pr('          A MULTIPLE QUANTUM WELL STRUCTURE');nl(3);

  2 -> number_of_layers
 endif;

 make_instance([SUPERLATTICE_FRAME]) -> superlattice_frame;
 superlattice_frame <- fill_the_frame;
 superlattice_frame <- logic_format;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
 enddefmethod;

endflavour;
```

```
;;;    25. SUPERLATTICE_WITH_A_FEW_LAYERS_FRAME OBJECT

flavour SUPERLATTICE_WITH_A_FEW_LAYERS_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5 slot6 slot7 slot8;

 ;;; This method allows you to fill the
 ;;; SUPERLATTICE_WITH_A_FEW_LAYERS_FRAME
 defmethod fill_the_frame;
 lvars layer_frame multiple_layer_peaks_frame interference_fringes_frame;
 lvars superlattice_frame;
 lvars check i quest1 quest4 quest7 quest8;
 lvars top_composition top_thickness bottom_composition bottom_thickness;

 "SUPERLATTICE_WITH_A_FEW_LAYERS_FRAME"->name_of_frame;
 8 -> number_of_slots;

 if x=1
 then
  nl(3);
  pr('          A SUPERLATTICE WITH A FEW LAYERS ');nl(2);
  pr('  FIRST DESCRIBE THE EXTRA LAYERS, THEN THE
       SUPERLATTICE');nl(3);

  '1. How many layers top the superlattice '->quest1;
  make_instance([QUESTION_SLOT])->slot1;
  0 -> check;
  while (check=0)
  do
   slot1 <- fill_slot("NUMBER_OF_TOP_LAYERS",quest1);nl(3);
   test_integer(slot1<-value) -> check
  endwhile;

  hd(slot1<-value) -> number_of_layers;
  newarray([1 ^number_of_layers]) -> top_composition;
  newarray([1 ^number_of_layers]) -> top_thickness;

  1 -> i;
  until i > number_of_layers
  do
   pr('2. Top Layer Number: ');pr(i);nl(2);

   make_instance([LAYER_FRAME]) -> layer_frame;
   layer_frame <- fill_the_frame;
   layer_frame <- logic_format;

   layer_composition -> top_composition(i);nl(2);
   layer_thickness -> top_thickness(i);nl(3);

   i+1 -> i
  enduntil;nl(1);

  make_instance([SLOT])->slot2;
  slot2 <- fill_slot("COMPOSITION_OF_LAYERS",top_composition);
  make_instance([SLOT])->slot3;
  slot3 <- fill_slot("THICKNESS_OF_LAYERS",top_thickness);nl(2);

  '3. How many layers beneath the superlattice '->quest4;
  make_instance([QUESTION_SLOT])->slot4;
  0 -> check;
  while (check=0)
  do
   slot4 <- fill_slot("NUMBER_OF_BOTTOM_LAYERS",quest4);nl(3);
   test_integer(slot4<-value) -> check
  endwhile;

  hd(slot4<-value) -> number_of_layers;
  newarray([1 ^number_of_layers]) -> bottom_composition;
  newarray([1 ^number_of_layers]) -> bottom_thickness;

  1 -> i;
  until i > number_of_layers
  do
   pr('4. Bottom Layer Number: ');pr(i);nl(2);

   make_instance([LAYER_FRAME]) -> layer_frame;
   layer_frame <- fill_the_frame;
   layer_frame <- logic_format;

   layer_composition -> bottom_composition(i);nl(2);
   layer_thickness -> bottom_thickness(i);nl(3);

   i+1 -> i
  enduntil;nl(1);

  make_instance([SLOT])->slot5;
  slot5 <-
  fill_slot("COMPOSITION_OF_BOTTOM_LAYERS",bottom_composition);
  make_instance([SLOT])->slot6;
  slot6 <-
  fill_slot("THICKNESS_OF_BOTTOM_LAYERS",bottom_thickness);nl(2)
 else
  make_instance([SLOT])->slot1;
  make_instance([SLOT])->slot2;
  make_instance([SLOT])->slot3;
  make_instance([SLOT])->slot4;
  make_instance([SLOT])->slot5;
  make_instance([SLOT])->slot6;

  slot1 <-
  fill_slot("NUMBER_OF_TOP_LAYERS",[^number_of_top_layers]);
  slot2 <- fill_slot("COMPOSITION_OF_LAYERS",top_layer_composition);
  slot3 <- fill_slot("THICKNESS_OF_LAYERS",top_layer_thickness);
  slot4 <-
  fill_slot("NUMBER_OF_BOTTOM_LAYERS",[^number_of_bottom_layers]);
  slot5 <-
   fill_slot("COMPOSITION_OF_BOTTOM_LAYERS",
   bottom_layer_composition);
  slot6 <-
  fill_slot("THICKNESS_OF_BOTTOM_LAYERS",bottom_layer_thickness);
 endif;

 make_instance([QUESTION_SLOT])->slot7;
 '5. How many PEAKS are there in the rocking curve ' -> quest7;
 0 -> check;
 while (check=0)
 do
  slot7 <- fill_slot("NUMBER_OF_PEAKS",quest7);nl(2);
  test_integer(slot7<-value) -> check
 endwhile;

 make_instance([Y_OR_N_QUESTION_SLOT])->slot8;
 '6. Do the layer peak(s) overlap with the other peaks (Y/N)' -> quest8;
 slot8 <- fill_slot("OVERLAP",quest8);nl(3);


 make_instance([MULTIPLE_LAYER_PEAKS_FRAME])
     -> multiple_layer_peaks_frame;
 multiple_layer_peaks_frame <- fill_the_frame;
 multiple_layer_peaks_frame <- logic_format;

 make_instance([SUPERLATTICE_FRAME]) -> superlattice_frame;
 superlattice_frame <- fill_the_frame;
 superlattice_frame <- logic_format;

 enddefmethod;

 ;;; Transform the frame into a logic-based format
 defmethod logic_format;
  hd(slot1<-value) -> number_of_top_layers;
  slot2<-value -> top_layer_composition;
  slot3<-value -> top_layer_thickness;
  hd(slot4<-value) -> number_of_bottom_layers;
  slot5<-value -> bottom_layer_composition;
  slot6<-value -> bottom_layer_thickness;
  hd(slot7<-value) -> number_of_peaks;
  assign_upper_case(slot8<-value) -> overlap_of_peaks;
 enddefmethod;

endflavour;


;;; ____________________________________________________


;;;  26. FRAME_SYSTEM OBJECT

flavour FRAME_SYSTEM;
 ivars leading_frame;
 ivars substrate_only_frame single_layer_frame single_graded_layer_frame;
 ivars multiple_layers_frame MQW_structure_frame;
 ivars superlattice_with_a_few_layers_frame superlattice_frame;


 defmethod ask_the_leading_question;
 0 -> type_of_structure_is_substrate_only;
 0 -> type_of_structure_is_single_layer;
 0 -> type_of_structure_is_multiple_layers;
 0 -> type_of_structure_is_MQW;
```

```
0 -> type_of_structure_has_additional_layers;
make_instance([LEADING_FRAME]) -> leading_frame;
leading_frame <- fill_the_frame;
leading_frame <- logic_format
enddefmethod;

;;;  FILL THE FRAME SYSTEM USING USER RESPONSES TO
       QUESTIONS
defmethod fill_the_frame_system(x);

if type_of_structure2 = 'additional layers'
then
  'superlattice capped with a few layers top and bottom' -> type_of_structure
endif;

if type_of_structure = 'substrate only'
then
  make_instance([SUBSTRATE_ONLY_FRAME]) -> substrate_only_frame;
  substrate_only_frame <- fill_the_frame;
  substrate_only_frame <- logic_format;

elseif type_of_structure = 'a single layer'
then
  make_instance([SINGLE_LAYER_FRAME]) -> single_layer_frame;
  single_layer_frame <- fill_the_frame;
  single_layer_frame <- logic_format;

elseif type_of_structure = 'multiple layers'
then
  make_instance([MULTIPLE_LAYERS_FRAME])->multiple_layers_frame;
  multiple_layers_frame <- fill_the_frame;
  multiple_layers_frame <- logic_format;

  'no additional layers' -> type_of_structure2;

elseif type_of_structure = 'MQW structure'
then
  make_instance([MQW_STRUCTURE_FRAME])->MQW_structure_frame;
  MQW_structure_frame <- fill_the_frame;
  MQW_structure_frame <- logic_format;

  'no additional layers' -> type_of_structure2;

elseif type_of_structure2 = 'additional_layers' or
  type_of_structure = 'superlattice capped with a few layers top and bottom'
then
  make_instance([SUPERLATTICE_WITH_A_FEW_LAYERS_FRAME])
          -> superlattice_with_a_few_layers_frame;
  superlattice_with_a_few_layers_frame <- fill_the_frame;
  superlattice_with_a_few_layers_frame <- logic_format;

  'MQW structure' -> type_of_structure;
  'additional layers' -> type_of_structure2;

else
  make_instance([SUPERLATTICE_FRAME]) -> superlattice_frame;
  superlattice_frame <- fill_the_frame;
  superlattice_frame <- logic_format;
  'MQW structure' -> type_of_structure;
  'no additional layers' -> type_of_structure2;
endif;

enddefmethod;

endflavour;



;;;  27. STRUCTURAL_PARAMETERS_FRAME OBJECT

flavour STRUCTURAL_PARAMETERS_FRAME isa FRAME;
 ivars slot1 slot2 slot3 slot4 slot5;

 ;;; This method allows you to fill
 ;;; the STRUCTURAL_PARAMETERS_FRAME
 defmethod fill_the_frame;
  lvars delta_theta theta v;
  lvars lamdba;
  lvars h k l;
  lvars Li Lj thetai thetaj;
  lvars delta_big_delta_theta big_delta_theta;
  lvars delta_theta_p gamma_h;

  "STRUCTURAL_PARAMETERS" -> name_of_frame;

  7 -> number_of_slots;

  if relaxed_layer = [Y]
  then
    0.8 -> relaxation
  endif;

  ;;; Derive the Experimental Mismatch: Fill slot1

  peak_splitting -> delta_theta;
  Bragg_angle -> theta;
  wavelength -> lambda;

  if experimental_mismatch > 0.1
  then
    if tan(theta)=0
    then
      theta+1 -> theta
    endif;

    -(delta_theta)*(1/tan(theta)) -> experimental_mismatch;

    ;;; Derive the Relaxed Mismatch: Fill slot2

    Poisson_ratio -> v;
    experimental_mismatch*(1-v)/(1+v) -> relaxed_mismatch;
  else
    0 -> relaxed_mismatch
  endif;

  make_instance([SLOT]) -> slot1;
  slot1<-fill_slot("EXPERIMENTAL_MISMATCH",experimental_mismatch);
  make_instance([SLOT]) -> slot2;
  slot2<-fill_slot("RELAXED_MISMATCH",relaxed_mismatch);

  ;;; Derive period of superlattice: Fill slot3

  satellite_order_1 -> Li;
  satellite_order_2 -> Lj;
  ((satellite_position_1/60) + theta) -> thetai;
  ((satellite_position_2/60) + theta) -> thetaj;

  if period_of_superlattice>0.125
  then
    if sin(thetai)=sin(thetaj)
    then
      thetai+1 -> thetai
    endif;
    (Li - Lj)*lambda / (2*(sin(thetai) - sin(thetaj))) -> period_of_superlattice;
  endif;
  clean(period_of_superlattice) -> period_of_superlattice;

  make_instance([SLOT]) -> slot3;
  slot3<-fill_slot("PERIOD_OF_SUPERLATTICE",period_of_superlattice);


  ;;; Derive period dispersion of superlattice: Fill slot4

  satellite_FWHM -> delta_big_delta_theta;
  abs(thetai-thetaj) -> big_delta_theta;

  if period_dispersion>0.125
  then
    if (cos(theta)*big_delta_theta)=0
    then
      theta+1 -> theta;
      big_delta_theta+1 -> big_delta_theta
    endif;
    (Li - Lj)*lambda*delta_big_delta_theta /
    ((cos(theta))*(big_delta_theta*big_delta_theta))
          -> period_dispersion;
  endif;
  clean(period_dispersion) -> period_dispersion;

  make_instance([SLOT]) -> slot4;
  slot4<-fill_slot("PERIOD_DISPERSION",period_dispersion);

  ;;; Derive layer thickness: Fill slot5

  if spacing_of_interference_fringes = 0
  then
    if wavelength=1.541 and type_of_structure='a single layer'
    then
      load thick.p;
```

```
  else
   0 -> thickness_of_layer
  endif;
 else
  spacing_of_interference_fringes -> delta_theta_p;
  cos(theta+90) -> gamma_h;

  if delta_theta_p=0
  then
   delta_theta_p+1 -> delta_theta_p
  endif;
  if sin(2*theta)=0
  then
   theta+1 -> theta
  endif;

  ;;; Change from arc seconds into radians
  (theta * 2 * pi) / (3600 * 360) -> theta;

  (lambda * gamma_h) / (delta_theta_p * sin(2*theta)) -> thickness_of_layer;

  ;;; Change from Angstroms to microns
  thickness_of_layer / (10 ** 4) -> thickness_of_layer;
  clean(thickness_of_layer) -> thickness_of_layer;

  if thickness_of_layer < 0
  then
   (-1)*thickness_of_layer -> thickness_of_layer
  endif;
 endif;

 make_instance([SLOT]) -> slot5;
 slot5<-fill_slot("THICKNESS_OF_LAYER",thickness_of_layer);

 clean(Bragg_angle) -> Bragg_angle;
enddefmethod;

;;; Print out the contents of the STRUCTURAL_PARAMETERS_FRAME
defmethod printself;
 lvars slot;

 nl(2);
 pr('_____________________________________________________________');
 nl(2);

 if type_of_structure = 'substrate only'
 then
  pr('  1. GENERAL INFORMATION');nl(2);
  pr('The Bragg angle is equal to ');pr(Bragg_angle);pr(' degrees.');nl(3);
  pr('Press ENTER to continue ...');readline() -> value;
  pr('_____________________________________________________________');
  nl(5);
  pr('  2. INFERRED CONSEQUENCES');nl(2);

  if (characteristic_curve > 0.1)
  then
   pr('RULE 13 HAS BEEN FIRED');nl(1);
   pr('This is the characteristic curve for a substrate only structure.');nl(1);
   pr('It is a curve of one peak only.');nl(2);
  else
   pr('RULE 13 HAS NOT BEEN FIRED');nl(1);
   pr('This is not the characteristic curve for a substrate only
        structure.');nl(1);
   pr('The characteristic curve is of one peak only.');nl(2)
  endif;

  if (strain_in_surface_layer_of_sample > 0.1)
  then
   pr('RULES 6 - 9  HAVE BEEN FIRED');nl(1);
   pr('There is ');pr(strain_in_surface_layer_of_sample);
   pr(' evidence of strain in surface layer of sample.');nl(1);
   pr('This is indicated by the asymmetry in the substrate peak.');nl(2)
  else
   pr('RULE 10 HAS BEEN FIRED');nl(1);
   pr('A change in thickness across the sample is not indicated');nl(2)
  endif;

  if (reference_crystal_is_different_to_substrate > 0.1)
  then
   pr('RULE 11 HAS BEEN FIRED');nl(1);
   pr('There is ');pr(reference_crystal_is_different_to_substrate);
   pr(' evidence that the reference crystal is different');nl(1);
   pr(' to the substrate.');nl(1);
   pr(' Alternatively the substrate may consist of subgrains.');nl(1);
   pr('This is inferred from the existence of more than one peak in the
        curve.');nl(2)
  endif;

  if (incorrect_experiment > 0.1)
  then
   pr('RULE 19 HAS BEEN FIRED');nl(1);
   pr('Your experiment is incorrect as there are NO PEAKS in the
        curve');nl(1);
   pr('This may be because');nl(1);
   pr('  a) You are doing the experiment incompetently and failing to find
        the peak;');nl(1);
   pr('OR');nl(1);
   pr('  b) The substrate peak is so bad that there is no detectable');nl(1);
   pr('  diffraction peak above the noise');nl(1);
   pr('OR');nl(1);
   pr('  c) You have looked at the wrong specimen. Maybe it is not
        crystalline');nl(2)
  endif;

  if misorientation_of_substrate > 0.2
  then
   pr('RULES 14 - 24 HAVE BEEN FIRED');nl(1);
   pr('There is ');pr(misorientation_of_substrate);
   pr(' evidence that the substrate is misoriented');nl(1);
   pr('This is indicated by broadening of the substrate peak');nl(2)
  endif;

  if (crystal_quality > 0.5)
  then
   pr('RULES 2 - 5 HAVE BEEN FIRED');nl(1);
   pr('The quality of this crystal is VERY GOOD');nl(1);
   pr('This is indicated by a single peak with little or no broadening.');nl(1);
   pr(' of the substrate peak.');nl(1);
   pr('Good crystal quality indicates that there is little misorientation or
        strain.');nl(2)
  elseif (crystal_quality < 0.3)
  then
   pr('RULE 1 HAS BEEN FIRED OR THERE IS NO SUBSTRATE
        PEAK');nl(1);
   pr('The quality of this crystal is BAD');nl(1);
   pr('This is indicated by extreme broadening of the substrate peak.');nl(1);
   pr('Bad crystal quality indicates that there is misorientation');nl(1);
   pr('of mosaic regions in the structure');nl(2)
  else
   pr('RULE 2 - 5 HAS BEEN FIRED');nl(1);
   pr('The quality of this crystal is REASONABLE');nl(1);
   pr('This is indicated by a small amount of broadening of the substrate
        peak.');nl(1);
   pr('There may be some misorientation or strain');nl(1);
   pr('in the structure ... and you should simulate for these.');nl(2)
  endif;
  nl(1);pr('Press ENTER to continue ...');readline() -> value;
  pr('_____________________________________________________________');
  nl(5);
  0.0 -> misoriented_or_mismatched_layer;

 elseif type_of_structure = 'a single layer' or type_of_structure = 'a single
        graded layer'
 then
  pr('  1. GENERAL INFORMATION');nl(2);
  pr('The Bragg angle is equal to ');pr(Bragg_angle);pr(' degrees.');nl(1);
   pr('The number of layers in this structure is
        ');pr(number_of_layers);pr('.');nl(2);
  pr('Press ENTER to continue ...');readline() -> value;
  pr('_____________________________________________________________');
  nl(2);
  pr('  2. CALCULATED VALUES');nl(2);
  pr('The experimental mismatch = ');pr(experimental_mismatch);pr('
        ppm.');nl(1);
  pr('The relaxed mismatch = ');pr(relaxed_mismatch);pr(' ppm.');nl(1);
  pr('The spacing of planes = ');pr(spacing_of_planes);nl(1);
  pr('The calculated thickness of the layer = ');pr(thickness_of_layer);nl(2);
  pr('Press ENTER to continue ...');readline() -> value;
  pr('_____________________________________________________________');
  nl(2);
  pr('  3. INFERRED CONSEQUENCES');nl(2);

  if (incorrect_experiment > 0.1)
  then
   pr('RULE 110 HAS BEEN FIRED');nl(1);
   pr('Your experiment is incorrect as there are NO PEAKS in the
        curve');nl(1);
   pr('This may be because');nl(1);
```

```
  pr('   a) You are doing the experiment incompetently and failing to find
          the peak;');nl(1);
  pr('OR');nl(1);
  pr('   b) The substrate peak is so bad that there is no detectable');nl(1);
  pr('      diffraction peak above the noise');nl(1);
  pr('OR');nl(1);
  pr('   c) You have looked at the wrong specimen. Maybe it is not
          crystalline');nl(2)
endif;

if (characteristic_curve > 0.1)
then
  pr('RULE 63 HAS BEEN FIRED');nl(1);
  pr('This is the characteristic curve for a single layer structure');nl(1);
  pr('This is a curve of two peaks');nl(2)
else
  pr('RULE 63 HAS NOT BEEN FIRED');nl(1);
  pr('This is not the characteristic curve for the single layer structure');nl(1);
  pr('The characteristic curve would have two peaks.');nl(2)
endif;

if (bending_of_substrate > 0.1)
then
  pr('RULES 2 - 5 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(bending_of_substrate);
  pr(' evidence of bending of the substrate in this structure.');nl(1);
  pr('This is indicated by broadening of the substrate peak.');nl(2)
else
  pr('RULE 6 HAS BEEN FIRED');nl(1);
  pr('There is NO evidence of bending of the substrate in this
          structure');nl(1);
  pr('Bending is usually indicated by broadening of the substrate peak');nl(2)
endif;

if (grading_of_the_layer > 0.1)
then
  pr('There is ');pr(grading_of_the_layer);
  pr(' evidence of grading of the layer in this structure.');nl(1);
  pr('Grading is usually indicated when the lattice parameters are');nl(1);
  pr(' quite close. That is, the mismatch is low.');nl(1);
  pr('When grading is indicated: check the width of the layer peak');nl(1);
  pr(' from the curve to where it tapers out near the background. This
          ');nl(1);
  pr('  will give an idea of what grading range to try in the
          simulation.');nl(1);
  pr('  Then fine tune by initially using a linear grade to get the
          approximate');nl(1);
  pr(' width of the peak ... followed by a nonlinear grade to obtain a
          better');nl(1);
  pr(' match between the detailed differences between the peaks ...');nl(2);
else
  pr('There is NO evidence of grading of the layer in this structure');nl(2);
endif;

if (evidence_of_mismatch > 0.1)
then
  pr('There is ');pr(evidence_of_mismatch);
  pr('  evidence of mismatch in this structure.');nl(1);
  pr('Please check the calculated result above');nl(1);
  pr(' Where the lattice parameters are NOT close, then');nl(1);
  pr(' mismatch rather than grading is indicated.');nl(1);
  pr(' The opposite is also the case.');nl(2);
endif;

if (change_in_lattice_parameter_with_depth > 0.1)
then
  pr('RULE 57 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(change_in_lattice_parameter_with_depth);
  pr(' evidence of a change in lattice parameter with depth.');nl(1);
  pr(' This is indicated by low visibility of interference fringes.');nl(2)
else
  pr('There is NO evidence of a change in lattice parameter with
          depth.');nl(2)
endif;

if (layer_is_present_in_the_substrate_peak > 0.1)
then
  pr('RULES 58 - 60 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(layer_is_present_in_the_substrate_peak);
  pr(' evidence that the layer is present in the substrate peak.');nl(1);
  pr(' This is indicated by asymmetry in the substrate peak.');nl(2)
endif;

if (peak_is_outside_the_scan_range > 0.1)
then
  pr('RULE 66 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(peak_is_outside_the_scan_range);
  pr(' evidence that a peak is outside the scan range.');nl(1);
  pr(' This is indicated by there being just a single peak.');nl(2)
endif;

if (layer_is_thick < layer_is_thin) and (layer_is_thin > 0.275)
then
  pr('There is ');pr(layer_is_thin);
  pr(' evidence that the layer is THIN');nl(1);
  pr(' This is indicated by the low intensity of the layer peak.');nl(1);
  pr(' The layer peak may disappear for very thin layers.');nl(2)
endif;

if(misoriented_or_mismatched_layer > 0.1)
then
  pr('RULE 67 OR 116-117 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(misoriented_or_mismatched_layer);
  pr(' evidence of a misoriented or mismatched layer');nl(1);
  pr(' This is indicated by the splitting of the layer peak.');nl(2);
  pr(' If there is tilt between the substrate and layer, then');nl(1);
  pr(' you will need FOUR rocking curves to characterize this tilt.');nl(1);
  pr(' Two symmetric and two asymmetric ones.');nl(1);
  pr('  First rotate the specimen by 180 degrees about the surface
          normal');nl(1);
  pr('  and calculate the tilt difference from 0 to 180 degree
          positions.');nl(1);
  pr(' Then rotate the specimen by 90 and 270 degrees and');nl(1);
  pr(' calculate tilt difference from 90 to 270 degree positions');nl(2);
  pr(' For strained layer systems (large mismatch) tilt optimization');nl(1);
  pr('  is often wasted time. Also, if you have a dispersive setup the
          tilt');nl(1);
  pr(' broadening is usually small compared to the dispersion broadening
          and');nl(1);
  pr(' it is again not worth doing tilt optimization');nl(2);
  pr('This rule may also have been fired because there is a');nl(1);
  pr(' multiple layer structure');nl(2)
endif;

if (simulation_or_calibration_chart_is_needed > 0.5)
then
  pr('RULES 71 OR 109 HAVE BEEN FIRED');nl(1);
  pr('There is evidence that more simulation or');nl(1);
  pr('a calibration chart is needed before the structural');nl(1);
  pr('parameters can be properly identified.');nl(1);
  pr('There is not enough peak separation between the substrate and');nl(1);
  pr(' layer peaks and so deductions may be off by about 20 percent.');nl(2)
endif;

if (layer_is_thin < layer_is_thick) and (layer_is_thick > 0.275)
then
  pr('There is ');pr(layer_is_thick);
  pr(' evidence that the layer is THICK.');nl(1);
  pr(' This is indicated by the high intensity of the layer peak.');nl(1);
  pr('  It is also indicated by a significant amount of interference
          fringes.');nl(2)
endif;

if (relaxation > 0.1)
then
  pr('RULES 97 - 99 OR RULE 108 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(relaxation);
  pr(' evidence that there is RELAXATION of the layer.');nl(1);
  pr(' If the layer is only partially coherent (it contains');nl(1);
  pr(' interface dislocations) it is said to be relaxed.');nl(2);
  pr(' In this case, the normal equation for mismatch is NOT valid.');nl(1);
  pr(' Hence it is necessary to measure the misfit parallel to the');nl(1);
  pr(' interface as well as perpendicular (e.g. 224 and 311 reflection');nl(1);
  pr(' indices). We then need to resolve the splitting to account for
          the');nl(1);
  pr(' inclination of the reflecting planes to the crystal surface.');nl(1);
  pr(' For this we need an asymmetric reflection.');nl(1);
  pr(' We represent a perpendicular and parallel mismatch and');nl(1);
  pr(' from these calculate the true mismatch.');nl(2)
endif;

if (misorientation_of_substrate > 0.5)
then
  pr('RULE 111 - 114 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(misorientation_of_substrate);
  pr(' evidence of misorientation of the substrate.');nl(1);
  pr(' Rotating specimen about 180 degrees will give a shift in');nl(1);
  pr(' the position of the Bragg peak. This will be exactly TWICE');nl(1);
```

```
  pr(' the misorientation angle between the reflecting plane and the');nl(1);
  pr(' specimen surface');nl(2)
 endif;

 if (crystal_quality > 0.5)
 then
  pr('The quality of this crystal is VERY GOOD');nl(1);
  pr(' This indicates that there is little misorientation or');nl(1);
  pr(' mismatch or grading ... check other indicators to verify this.');nl(2)
 elseif (crystal_quality < 0.3)
 then
  pr('The quality of this crystal is BAD');nl(1);
  pr(' This indicates that there is AN AMOUNT of misorientation or');nl(1);
  pr(' mismatch or grading ... check other indicators to verify this.');nl(2)
 else
  pr('RULE 1 HAS BEEN FIRED');nl(1);
  pr('The quality of this crystal is REASONABLY GOOD.');nl(2);
 endif;

 0.0 -> layers_are_thick;
 0.0 -> grading_occurs_through_AB_layers;
 0.0 -> grading_or_dispersion_of_layer_thicknesses;

 if (grading_of_the_layer > 0.1)
 then
  pr('There is evidence that the layer may be graded.');nl(2);
  pr('What is a possible grading distance as a first estimate ');readline() ->
       grading_distance;nl(2);
 endif;nl(2);
 nl(1);pr('Press ENTER to continue ...');readline() -> value;
 pr('_____');
 nl(3);

elseif type_of_structure = 'non-graded multiple layers' or type_of_structure
    = 'graded multiple layers' or type_of_structure = 'multiple layers'
then
 pr('  1. GENERAL INFORMATION');nl(2);
 pr('The Bragg angle is equal to ');pr(Bragg_angle);pr(' degrees.');nl(2);
 pr('The number of layers in this structure is ');pr(number_of_layers);nl(1);
 pr('The number of blocks in this structure is ');pr(number_of_blocks);nl(2);
 pr('Press ENTER to continue ...');readline() -> value;
 pr('_____');
 nl(2);
 pr('  2. CALCULATED VALUES');nl(2);
 pr('The experimental mismatch = ');pr(experimental_mismatch);nl(1);
 pr('The relaxed mismatch = ');pr(relaxed_mismatch);nl(1);
 pr('The spacing of planes = ');pr(spacing_of_planes);nl(2);
 pr('Press ENTER to continue ...');readline() -> value;
 pr('_____');
 nl(2);
 pr('  3. INFERRED CONSEQUENCES');nl(3);

 if (incorrect_experiment > 0.1)
 then
  pr('RULE 13 HAS BEEN FIRED');nl(1);
  pr('Your experiment is incorrect as there are NO PEAKS in the
       curve');nl(1);
  pr('This may be because');nl(1);
  pr('  a) You are doing the experiment incompetently and failing to find
       the peak;');nl(1);
  pr('OR');nl(1);
  pr('  b) The substrate peak is so bad that there is no detectable');nl(1);
  pr('    diffraction peak above the noise');nl(1);
  pr('OR');nl(1);
  pr('  c) You have looked at the wrong specimen. Maybe it is not
       crystalline');nl(2)
 endif;

 if (simulation_or_calibration_chart_is_needed > 0.1)
 then
  pr('RULES 8 - 11 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(simulation_or_calibration_chart_is_needed);
  pr(' evidence that more simulation or');nl(1);
  pr(' a calibration chart is needed before the structural');nl(1);
  pr(' parameters can be properly identified.');nl(1);
  pr(' The large number of interference fringes suggest that');nl(1);
  pr(' simulation is needed to identify the features of this structure.');nl(2)
 endif;

 if (layers_are_thick > 0.275)
 then
  pr('RULE 6 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(layers_are_thick);
  pr(' evidence that the layers are THICK.');nl(1);
  pr(' This is because each layer has an identifiable peak.');nl(2)
 else
  pr('There is evidence that at least some of the layers are THIN.');nl(1);
  pr(' This is indicated by the fact that not EVERY layer has an identifiable
       peak.');nl(2)
 endif;

 if (there_are_hidden_layers_somewhere > 0.1)
 then
  pr('RULE 7 OR RULES 19 - 22 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(there_are_hidden_layers_somewhere);
  pr(' evidence that there are thin layers somewhere');nl(1);
  pr('in the structure.');nl(1);
  pr('This is indicated either by interference fringes or');nl(1);
  pr('because the number of peaks does not correspond to the layers');nl(2);
 endif;

 if (relaxation > 0.2)
 then
  pr('RULES 1 - 4 OR RULE 30 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(relaxation);
  pr(' evidence that there is RELAXATION of the layer.');nl(1);
  pr(' If the layer is only partially coherent (it contains');nl(1);
  pr(' interface dislocations) it is said to be relaxed.');nl(2);
  pr(' In this case, the normal equation for mismatch is NOT valid.');nl(1);
  pr(' Hence it is necessary to measure the misfit parallel to the');nl(1);
  pr(' interface as well as perpendicular (e.g. 224 and 311 reflection');nl(1);
  pr(' indices). We then need to resolve the splitting to account for
       the');nl(1);
  pr(' inclination of the reflecting planes to the crystal surface.');nl(1);
  pr(' For this we need an asymmetric reflection.');nl(1);
  pr(' We represent a perpendicular and parallel mismatch and');nl(1);
  pr(' from these calculate the true mismatch.');nl(2)
 endif;

 if (evidence_of_mismatch > 0.1)
 then
  pr('RULE 29 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(evidence_of_mismatch);
  pr(' evidence of mismatch in this structure.');nl(1);
  pr('This is indicated by the number of peaks not corresponding to');nl(1);
  pr(' the number of layers.');nl(2)
 endif;

 if (there_are_thin_layers_at_the_interfaces > 0.1)
 then
  pr('RULE 31 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(there_are_thin_layers_at_the_interfaces);
  pr(' evidence that there are thin layers present');nl(1);
  pr(' at the interfaces between layers.');nl(1);
  pr('This is indicated by a split substrate peak.');nl(2)
 endif;

 if (evidence_of_interferometer_structure > 0.1)
 then
  pr('RULES 24 - 27 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(evidence_of_interferometer_structure);
  pr(' evidence of an interferometer structure.');nl(1);
  pr('This is a structure with a very thin layer between two');nl(1);
  pr(' thick layers. This is indicated by extra peaks in the ');nl(1);
  pr(' curve and movement of peaks with respect to the substrate
       peak.');nl(2)
 endif;

 if (misorientation_of_substrate > 0.1)
 then
  pr('There is ');pr(misorientation_of_substrate);
  pr(' evidence of misorientation of the substrate.');nl(1);
  pr(' Rotating specimen about 180 degrees will give a shift in');nl(1);
  pr(' the position of the Bragg peak. This will be exactly TWICE');nl(1);
  pr(' the misorientation angle between the reflecting plane and the');nl(1);
  pr(' specimen surface');nl(2)
 endif;

 nl(1);pr('Press ENTER to continue ...');readline() -> value;
 pr('_____');
 nl(5);
 0.0 -> misoriented_or_mismatched_layer;

elseif type_of_structure = 'MQW structure' or type_of_structure =
  'superlattice capped with a few layers top and bottom' or
  type_of_structure = 'graded superlattice'
then
  pr('  1. GENERAL INFORMATION');nl(2);
```

```
pr('The Bragg angle is equal to ');pr(Bragg_angle);pr(' degrees.');nl(2);
pr('The number of layers in this structure is ');pr(number_of_layers);nl(1);
pr('The number of blocks in this structure is ');pr(number_of_blocks);nl(2);
pr('Press ENTER to continue ...');readline() -> value;
pr('_____');
nl(2);
pr('  2. CALCULATED VALUES');nl(2);
pr('The experimental mismatch = ');pr(experimental_mismatch);nl(1);
pr('The relaxed mismatch = ');pr(relaxed_mismatch);nl(1);
pr('The period of the superlattice = ');pr(period_of_superlattice);nl(1);
  pr('The period dispersion in the superlattice =
      ');pr(period_dispersion);nl(2);
pr('Press ENTER to continue ...');readline() -> value;
pr('_____');
nl(2);
pr('  3. INFERRED CONSEQUENCES');nl(3);

if (incorrect_experiment > 0.1)
then
  pr('RULE 71 HAS BEEN FIRED');nl(1);
  pr('Your experiment is incorrect as there are NO PEAKS in the
      curve');nl(1);
  pr('This may be because');nl(1);
  pr('  a) You are doing the experiment incompetently and failing to find
      the peak;');nl(1);
  pr('OR');nl(1);
  pr('  b) The substrate peak is so bad that there is no detectable');nl(1);
  pr('     diffraction peak above the noise');nl(1);
  pr('OR');nl(1);
  pr('  c) You have looked at the wrong specimen. Maybe it is not
      crystalline');nl(2)
endif;

if (characteristic_curve > 0.2)
then
  pr('There is ');pr(characteristic_curve);
  pr(' evidence that this is the characteristic curve for');nl(1);
  pr('a superlattice structure:');nl(1);
  pr('  That is a curve with one substrate peak');nl(1);
  pr('  one zero order peak and satellite peaks arranged');nl(1);
  pr('  symmetrically about the zero order peak.');nl(2)
else
  pr('This is not the characteristic curve for the superlattice
      structure:');nl(1);
  pr(' The characteristic curve should contain a substrate peak');nl(1);
  pr(' a zero order peak and satellite peaks arranged symmetrically');nl(1);
  pr(' about the zero order peak.');nl(2)
endif;

if (simulation_or_calibration_chart_is_needed > 0.5)
then
  pr('RULE 61 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(simulation_or_calibration_chart_is_needed);
  pr(' evidence that more simulation or');nl(1);
  pr('a calibration chart is needed before the structural');nl(1);
  pr('parameters can be properly identified.');nl(1);
  pr('This is because the superlattice thickness is low');nl(1);
  pr(' (below 0.5 microns thick)');nl(2)
endif;

if (layers_are_thick > layers_are_thin) and (layers_are_thick > 0.275)
then
  if satellite_spacing_of_peaks = 0.0
  then
  else
    pr('RULES 11 - 14 HAVE BEEN FIRED');nl(1);
    pr('There is ');pr(layers_are_thick);
    pr(' evidence that the layers are THICK.');nl(1);
    pr(' This is either because there is little separation between the
      satellites.');nl(2)
  endif;
endif;

if (layers_are_thin > layers_are_thick) and (layers_are_thin > 0.275)
then
  pr('RULES 6 - 9 OR 65 - 68 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(layers_are_thin);
  pr(' evidence that the layers are THIN.');nl(1);
  pr(' This is either because there is large separation between the satellite
      peaks,');nl(1);
  pr(' or because there are large numbers of interference fringes.');nl(2)
endif;

if (grading_occurs_through_AB_layers > 0.1)
then
  pr('RULES 22 - 25 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(grading_occurs_through_AB_layers);
  pr(' evidence that grading occurs through');nl(1);
  pr('  the AB layers of this superlattice.');nl(1);
  pr('This is indicated because the visibility of satellites is low.');nl(2)
endif;

if (grading_or_dispersion_of_layer_thicknesses > 0.5)
then
  pr('RULES 17 - 20 OR 28 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(grading_or_dispersion_of_layer_thicknesses);
  pr(' evidence that grading or dispersion of.');nl(1);
  pr('of layer thicknesses occurs.');nl(1);
  pr(' This is indicated through broadening of higher order satellites.');nl(1);
  pr(' or through asymmetry of plus and minus satellites. ');nl(2)
endif;

if (relaxation > 0.1)
then
  pr('There is ');pr(relaxation);
  pr(' evidence that there is RELAXATION of the layer.');nl(1);
  pr(' If the layer is only partially coherent (it contains');nl(1);
  pr(' interface dislocations) it is said to be relaxed.');nl(2);
  pr(' In this case, the normal equation for mismatch is NOT valid.');nl(1);
  pr(' Hence it is necessary to measure the misfit parallel to the');nl(1);
  pr(' interface as well as perpendicular (e.g. 224 and 311 reflection');nl(1);
  pr(' indices). We then need to resolve the splitting to account for
      the');nl(1);
  pr(' inclination of the reflecting planes to the crystal surface.');nl(1);
  pr(' For this we need an asymmetric reflection.');nl(1);
  pr(' We represent a perpendicular and parallel mismatch and');nl(1);
  pr(' from these calculate the true mismatch.');nl(2)
endif;

if (large_overall_layer_thickness > 0.1)
then
  pr('RULE 27 HAS BEEN FIRED');nl(1);
  pr('There is ');pr(large_overall_layer_thickness);
  pr(' evidence that there is a large overall layer thickness in');nl(1);
  pr(' this structure. This is indicated by subsidiary interference effects ');
  pr(' on the satellite peaks.');nl(2)
endif;

if (layers_are_not_uniform > 0.1)
then
  pr('RULES 54 - 57 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(layers_are_not_uniform);
  pr(' evidence that the layers are not uniform.');nl(1);
  pr('This is indicated by the low visibility of satellites.');nl(2)
endif;

if (misorientation_of_substrate > 0.5)
then
  pr('RULES 73 - 75 HAVE BEEN FIRED');nl(1);
  pr('There is ');pr(misorientation_of_substrate);
  pr(' evidence of misorientation of the substrate.');nl(1);
  pr(' Rotating specimen about 180 degrees will give a shift in');nl(1);
  pr(' the position of the Bragg peak. This will be exactly TWICE');nl(1);
  pr(' the misorientation angle between the reflecting plane and the');nl(1);
  pr(' specimen surface');nl(2)
endif;
nl(1);pr('Press ENTER to continue ...');readline() -> value;
0.0 -> misoriented_or_mismatched_layer;

endif;
nl(5);
pr('_____');
nl(2);
pr('       INPUTS FOR DATA ENTRY WINDOWS IN THE CREATE
      SCREENS IN RADS');nl(2);
pr('_____');
nl(2);
pr('First Window: CONTROL LINE PARAMETERS');nl(2);
pr('    Include reference crystal :      Using the reference crystal
      increases');nl(1);
pr('                                the time of the simulation but results');nl(1);
pr('                                in a more accurate graph.');nl(2);
pr('  Choose state of polarisation : SAME AS EXPERIMENT');nl(2);
pr('  Wavelength : ');pr(wavelength);pr(' Angstroms.');nl(2);
pr('  Reflection indices :');nl(1);
pr('                H = ');pr(h_reflection_index);nl(1);
pr('                K = ');pr(k_reflection_index);nl(1);
pr('                L = ');pr(l_reflection_index);nl(2);
```

```
pr('  Scan range :  Start  = ');pr(start_of_scan_range);nl(1);
pr('              Finish = ');pr(end_of_scan_range);nl(2);
pr('  The size of the scan step must also be entered.');nl(2);
pr('_____');
nl(2);
pr('Second window: SUBSTRATE SELECTION');nl(2);
pr('  Substrate material: ');pr(substrate_material);nl(1);
nl(1);pr('Press ENTER to continue ...');readline() -> value;nl(1);
pr('_____');
nl(2);
pr('Third window: REFLECTION GEOMETRY');nl(2);
pr('  Selected reflection orientation : ');pr(reflection_orientation);nl(2);
pr('  Surface normal indices :');nl(1);
pr('                        H = ');pr(h_surface_normal_index);nl(1);
pr('                        K = ');pr(k_surface_normal_index);nl(1);
pr('                        L = ');pr(l_surface_normal_index);nl(2);
pr('  Wafer misorientation : ');
if (misorientation_of_substrate > 0.1 or misoriented_or_mismatched_layer >
   0.1 or crystal_quality = 0.1)
then
  pr('There is evidence of some misorientation');nl(1);
  pr('                  Check the inferred consequences above ...')
else
  pr('There is no obvious evidence of misorientation')
endif;nl(2);
pr('_____');
nl(2);
pr('Fourth window: REFERENCE SELECTION');nl(2);
pr('  THIS IS THE SAME AS THE EXPERIMENT');nl(2);
pr('      The reference crystal is usually the same material as the
          substrate.');nl(1);
nl(1);pr('Press ENTER to continue ...');readline() -> value;nl(1);
pr('_____');
nl(2);
pr('Fifth window: OVERLAYER DEFINITION');nl(2);
pr('  Number of layers : ');pr(number_of_layers);nl(2);
pr('_____');
nl(2);
pr('Sixth window: LAYER CONTROL LINE');nl(2);
1 -> x;
if number_of_peaks_is_one = 1
then
  number_of_layers+1 -> x
endif;
until x > number_of_layers
do
  if number_of_layers > 1
  then
    layer_frame(x)<-slot4 -> slot;
    hd(slot<-value) -> layer_thickness
  endif;
  pr('Layer Number ');pr(x);nl(2);
  pr('  Layer thickness = ');pr(layer_thickness);pr(' microns.');nl(1);
  if number_of_layers = 1
  then
    pr('  Calculated layer thickness = ');pr(thickness_of_layer);
    pr(' microns');nl(2);
  endif;
  pr('  Lamellae/layer : Depends on whether composition and
     strain changes');nl(1);
  pr('                   with depth also used when grading occurs ...');nl(1);
  if not(grading_of_the_layer = 0.0)
  then
    if grading_of_the_layer > 0.1
    then
      pr('                   As there is evidence of grading then
            use 10 lamellae.');nl(2);
    else
      pr('                   As there is NO evidence of grading then
            use 1 lamella.');nl(2);
    endif;
  endif;
  pr('  Relaxation : Depends on whether MISMATCH is large and
            whether layer');nl(1);
  pr('               or layers are thick. In this case');nl(1);
  if (relaxation > 0.8)
  then
    pr('            Choose 100 percent relaxation.');nl(2);
  elseif (relaxation > 0.5 or relaxed_mismatch > 500 or relaxed_mismatch
     < -500 or layer_is_thick > 0.8 or layers_are_thick > 0.8)
  then
    pr('            Choose relaxation greater than 50 percent and');nl(1);
    pr('            vary the amount of relaxation until peak ');nl(1);
    pr('            separation is OK.');nl(2);
  elseif (relaxation > 0.2 or relaxed_mismatch > 100 or relaxed_mismatch
     < -100 or layer_is_thick > 0.3 or layers_are_thick > 0.3)
  then
    pr('            Choose relaxation greater than 20 percent.');nl(2);
  else
    pr('            Choose ZERO relaxation.');nl(2)
  endif;

  pr('  Total number of blocks : ');pr(number_of_blocks);nl(1);
  nl(1);pr('Press ENTER to continue ...');readline() -> value;nl(1);
  pr('_____');
  nl(2);
  pr('Seventh window: LAYER DEFINITION');nl(2);
  if number_of_layers > 1
  then
    layer_frame(x)<-slot3 -> slot;
    slot<-value -> layer_chemistry
  endif;
  pr('  Select layer chemistry : ');pr(layer_chemistry);nl(2);
  if number_of_layers > 1
  then
    layer_frame(x)<-slot2 -> slot;
    slot<-value -> layer_material
  endif;
  pr('  Material for current layer : ');pr(layer_material);nl(2);
  pr('  Constants defining X composition :');nl(1);
  pr('  These fields are used when simulating grading ...');nl(2);
  if (grading_of_the_layer > 0.1 or grading_occurs_through_AB_layers
      > 0.1 or grading_or_dispersion_of_layer_thicknesses > 0.1)
  then
    pr('    There is evidence of grading so choose');nl(1);
    pr('      A = 0; B = some constant; C = some constant ... for
              linear grading');nl(1);
    if (type_of_structure='a single layer' or type_of_structure='a single
              graded layer')
    then
      pr('    At t=0 X=C; t=');pr(grading_distance);
      pr(' X=A*');pr(grading_distance);pr('+C');nl(1);
      pr('    This depends on the proportions of each material
            in the alloy.');nl(2);
    endif;
    pr('                    or');nl(1);
    pr('      A = some constant; B = 0 or some constant; C = some
                constant ... for quadratic grading');nl(2);
    pr('      Similar constraints used to define Y composition
                for quaternary');nl(2);
  else
    pr('  There is NO evidence of grading so choose');nl(1);
    pr('    A = 0; B = 0; C = some constant.');nl(1);
  endif;nl(2);
  pr('Press ENTER to continue ...');readline() -> value;nl(1);
  pr('_____');
  nl(2);
  pr('Eighth window: STRAIN DEFINITION');nl(2);
  pr('  This window is used for additional strain definition.');nl(1);
  pr('  Strain is additional to that defined by material composition.');nl(2);
  pr('  This window is here for when you have something
        like mechanical');nl(1);
  pr('  damage or ion implantation etc and you can use this window
        to ');nl(1);
  pr('  simulate composition change by means of a strain change ...');nl(2);
  pr('  For relaxed mismatch:');nl(2);
  pr('  ');pr(relaxed_mismatch);pr(' ppm = At2 + Bt + C where t is
        height above');nl(1);
  pr('                    bottom of layer.');nl(2);
  pr('  You can also use experimental mismatch in this equation.');nl(2);
  pr(' DEFINE A B and C ACCORDINGLY ... it is up to the
        user to');nl(1);
  pr('  enter sensible values.');nl(1);
  nl(1);pr('Press ENTER to continue ...');readline() -> value;nl(1);
  pr('_____');
  nl(2);
  x+1 -> x
 enduntil;
 nl(3);
 pr('_____');
 nl(10);
enddefmethod;

endflavour;


;;; _____
```

```
;;; 28. FUZZY_VARIABLE OBJECT
flavour FUZZY_VARIABLE;
 ivars input_value fuzzy_set;
 ivars membership;

 defmethod calculate_value_of_premise(value, A);
  lvars value A size;
  lvars count i start finish halfway midpoint;

  round(length(A) / 2.0) -> size;
  newarray([1 ^size]) -> membership;

  1 -> count;
  1 -> i;
  until i>10
  do
   A(i) -> start;
   i+1 -> i;
   A(i) -> finish;

   (finish - start)/2.0 -> halfway;
   start + halfway -> midpoint;

   if value < midpoint
   then
    (value - start) / (midpoint - start) -> membership(count)
   else
    ((midpoint - value) / (finish - midpoint)) + 1.0 -> membership(count)
   endif;

   if membership(count) > 1.0 or membership(count) < 0.0
   then
    0.0 -> membership(count)
   endif;

   count+1 -> count;
   i+1 -> i
  enduntil;

  return(membership);

 enddefmethod;


 defmethod calculate_value_of_conclusion(fuzzy_variable_number);
  lvars fuzzy_variable_number membership_function;

  lvars i B values size defuzzify Yj;
  lvars temp_list j value count y_value k;
  lvars start finish halfway midpoint top_value bottom_value;
  lvars centroid_defuzzification;

  fuzzy_variable_number -> i;
  fuzzy_variable(i,7) -> B;

  [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0] -> values;

  length(B) / 2 -> size;
  newarray([1 ^size]) -> defuzzify;

  length(values) -> size;
  newarray([1 ^size 1 2]) -> Yj;

  values -> temp_list;
  0 -> j;
  until temp_list=[]
  do
   hd(temp_list) -> value;
   j+1 -> j;
   1 -> count;
   0.0 -> y_value;

   until count > 5
   do
    if fuzzy_variable(i, count) > 0
    then
     (2 * count) - 1 -> k;
     B(k) -> start;
     k+1 -> k;
     B(k) -> finish;

     (finish - start) / 2 -> halfway;
     start + halfway -> midpoint;

     if value < start or value > finish
     then
      0.0 -> defuzzify(count)
     else
      if value < midpoint
      then
       (value - start) / (midpoint - start) -> defuzzify(count)
      else
       ((midpoint - value) / (finish - midpoint)) + 1 -> defuzzify(count)
      endif;
     endif;

     if defuzzify(count) > fuzzy_variable(i, count)
     then
      fuzzy_variable(i, count) -> defuzzify(count)
     endif;
    else
     0.0 -> defuzzify(count)
    endif;

    max(defuzzify(count), y_value) -> y_value;

    count+1 -> count

   enduntil;

   value -> Yj(j,1);
   y_value -> Yj(j,2);

   tl(temp_list) -> temp_list;

  enduntil;

  0.0 -> top_value;
  0.0 -> bottom_value;
  1 -> j;
  until j > length(values)
  do
   Yj(j,1) * Yj(j,2) + top_value -> top_value;
   Yj(j,2) + bottom_value -> bottom_value;
   j+1 -> j;
  enduntil;

  if bottom_value = 0
  then
   0.0 -> centroid_defuzzification
  else
   top_value / bottom_value -> centroid_defuzzification;
   clean(centroid_defuzzification)
  endif;

  clean(centroid_defuzzification) -> centroid_defuzzification;
  centroid_defuzzification -> fuzzy_variable(i,6);

 enddefmethod;

endflavour;

;;; FUNCTIONS DEFINING INEQUALITIES FOR THE RULE OBJECT

define is_equal_to(x,y) -> r;
 if x=y
  then true -> r
  else false -> r
 endif;
enddefine;

define is_not_equal_to(x,y) -> r;
 if x=y
  then false -> r
  else true -> r
 endif;
enddefine;

define greater_than(x,y) -> r;
 if x > y
  then true -> r
  else false -> r
 endif;
enddefine;


define less_than(x,y) -> r;
 if x < y
```

```
    then true -> r
    else false -> r
  endif;
enddefine;


define greater_than_or_equal_to(x,y) -> r;
  if (x > y or x=y)
    then true -> r
    else false -> r
  endif;
enddefine;


define less_than_or_equal_to(x,y) -> r;
  if (x < y or x=y)
    then true -> r
    else false -> r
  endif;
enddefine;

define is_assigned_the_value(x,y);
  y -> x
enddefine;


;;;    29. RULE OBJECT

flavour RULE;
  ivars premise consequence;

  defmethod use_rule(head, tail);
    lvars head tail truth_value_of_premise;
    lvars count i check Bj fired A;

    head -> premise;
    tail -> consequence;

    1.0 -> truth_value_of_premise;
    0 -> fired;

    if length(premise) > 3
    then
      1 -> count;
      1 -> i;
      until i > length(premise)
      do
        if premise(i) = 1
        then
          apply(premise(i+1), premise(i+3), premise(i+2)) -> check;
          if check = true
          then
            1.0 -> check
          else
            0.0 -> check
          endif;
          i+5 -> i
        else
          make_instance([FUZZY_VARIABLE]) -> A;
          A <- calculate_value_of_premise(premise(i+1), premise(i+4)) -> Bj;
          cancel A;

          if premise(i+3) = 'EXTREME'
          then
            Bj(5) -> check
          elseif premise(i+3) = 'VERY'
          then
            Bj(4) -> check
          elseif premise(i+3) = 'FAIRLY'
          then
            Bj(3) -> check
          elseif premise(i+3) = 'SOME'
          then
            Bj(2) -> check
          else
            Bj(1) -> check
          endif;

          i+6 -> i
        endif;

        min(truth_value_of_premise, check) -> truth_value_of_premise;
      enduntil;
    endif;

    consequence(1) -> i;

    if consequence(4) = 'EXTREME'
    then
      max(truth_value_of_premise, fuzzy_variable(i,5)) -> fuzzy_variable(i,5)
    elseif consequence(4) = 'VERY'
    then
      max(truth_value_of_premise, fuzzy_variable(i,4)) -> fuzzy_variable(i,4)
    elseif consequence(4) = 'FAIRLY'
    then
      max(truth_value_of_premise, fuzzy_variable(i,3)) -> fuzzy_variable(i,3)
    elseif consequence(4) = 'SOME'
    then
      max(truth_value_of_premise, fuzzy_variable(i,2)) -> fuzzy_variable(i,2)
    else
      max(truth_value_of_premise, fuzzy_variable(i,1)) -> fuzzy_variable(i,1)
    endif;

    if truth_value_of_premise > 0.0
    then
      1 -> fired
    endif;

    return(fired)

  enddefmethod;

endflavour;

;;;-----------------------------------------------------------------------


;;; 30. FUNCTION TO ADD RULES TO A CONNECTION MATRIX
;;;     (This function uses recursion)

define add_rules_to_CM(matrix,list_of_rules);
  lvars rule_number row column;

  if list_of_rules=[]
  then
  else
    hd(list_of_rules) -> rule_number;
    (rule_number * 2)-1 -> row;
    (rule_number * 2) -> column;

    1 -> matrix(row,column);
    add_rules_to_CM(matrix,tl(list_of_rules))
  endif;
enddefine;


;;; 31. CONNECTION_MATRIX OBJECT

flavour CONNECTION_MATRIX;
  ivars matrix_name matrix cred_matrix list_of_rules credibility_weight
history_of_decisions history_of_incs_decs;

  ;;; LIST OF METHODS
  ;;;
  ;;; 1) create_a_connection_matrix(name, rules);
  ;;; 2) assign_credibility_weight(number);
  ;;; 3) scalar_multiply();
  ;;; 4) access_cred_matrix(i,j);
  ;;; 5) update_history(outcome);
  ;;; 6) check_success(percentage);
  ;;; 7) change_credibility_weight(inc_or_dec);
  ;;; 8) assign_histories(number);
  ;;; 9) save_to_file(x);
  ;;; 10) retrieve_from_file(x);
  ;;; 11) store_angst;
  ;;; 12) printself;


  ;;; METHOD: 1) create_a_connection_matrix(name, ruleset);
  ;;; Method to create a connection matrix
  defmethod create_a_connection_matrix(name, ruleset);
    lvars name ruleset size_of_matrix;

    name -> matrix_name;
    ruleset -> matrix;

    number_of_rules * 2 -> size_of_matrix;
    newarray([1 ^size_of_matrix 1 ^size_of_matrix],0) -> cred_matrix;
  enddefmethod;
```

```
;;; METHOD: 2) assign_credibility_weight(number);
;;; This method assigns a credibility weight to an expert
defmethod assign_credibility_weight(number);
  !vars number;

  number -> credibility_weight
enddefmethod;


;;; METHOD: 3) scalar_multiply();
;;;  This method scalar multiplies the expert's connection matrix by the
credibility weight
defmethod scalar_multiply;
  !vars size_of_matrix i j;

  1 -> i;
  1 -> j;
  number_of_rules * 2 -> size_of_matrix;

  until i > size_of_matrix
  do
    until j > size_of_matrix
    do
      matrix(i,j) * credibility_weight -> cred_matrix(i,j);
      j+1 -> j
    enduntil;

    1 -> j;
    i+1 -> i

  enduntil;

enddefmethod;


;;; METHOD: 4) access_cred_matrix(i,j);
;;; This method allows access to the value contained at specific indices of the
;;; connection matrix. It is used by the Combined Matrix object (see below).
defmethod access_cred_matrix(i,j);
  !vars i j;

  cred_matrix(i,j)
enddefmethod;


;;; METHOD: 5) update_history(outcome);
;;; This method updates the history of expert system decisions for this expert
;;; Each new outcome of decision is put at the END of the list
defmethod update_history(outcome);
  !vars outcome Number_of_decisions;

  20 -> Number_of_decisions;
  if length(history_of_decisions) < Number_of_decisions
  then
    history_of_decisions <> [^outcome] -> history_of_decisions
  else
    tl(history_of_decisions) <> [^outcome] -> history_of_decisions
  endif;

enddefmethod;


;;; METHOD: 6) check_success(percent1, percent2);
;;; This method decides whether increments or decrements are to be
;;; made to the credibility weight.
;;; This depends on the recorded history of expert system decisions.
;;; It uses the next method in order to implement any change to the weighting.
defmethod check_success(percent1, percent2);
  !vars percent1 percent2 SPECIAL_VARIABLE;
  !vars temp_list outcome success failure inc_or_dec;

  0 -> SPECIAL_VARIABLE;

  history_of_decisions -> temp_list;
  0 -> success;
  0 -> failure;

  ;;; Count the number of successful and unsuccessful decisions recorded
  ;;; in the history_of_decisions
  until temp_list = []
  do
    hd(temp_list) -> outcome;
    if outcome = 1
    then
      success + 1 -> success
    else
      failure + 1 -> failure
    endif;

    tl(temp_list) -> temp_list

  enduntil;

  ;;; If this expert has consistently successful rules
  ;;; then the credibility weighting is incremented.
  if (success > percent1 or success = percent1)
  then
    1 -> inc_or_dec;
    ^change_credibility_weight(inc_or_dec);
    1 -> SPECIAL_VARIABLE;
    1 -> angst;
    ^store_angst;
  endif;

  ;;; If this expert has consistently unsuccessful rules
  ;;; then the credibility weighting is decremented.
  if (failure > percent2 or failure = percent2)
  then
    0 -> inc_or_dec;
    ^change_credibility_weight(inc_or_dec);
    1 -> SPECIAL_VARIABLE;
    1 -> angst;
    ^store_angst;
  endif;

  return(SPECIAL_VARIABLE)

enddefmethod;


;;; METHOD: 7) change_credibility_weight(inc_or_dec);
;;; This method implements the mathematical formula:
;;;
;;; Inc = (0.5 modulus(0.5 - W)) * f(p,q) * 1/kappa
;;;
;;; Dec = (0.5 modulus(0.5 - W)) * f(1/p,1/q) * 1/kappa
;;;
;;; where  f(p,q) = Min((p+1)/(q+1)) / Max((p+1)/(q+1))
;;;
;;; This formula is used to calculate the size of increments and decrements to
;;; the credibility weight
defmethod change_credibility_weight(inc_or_dec);
  !vars inc_or_dec;

  ;;; Variables used in the above mathematical formula
  !vars W epsilon p q f g;

  ;;; Temporary variables used to define the above set of variables
  !vars temp_list i P_positive P_negative Q_positive Q_negative Inc;

  ;;; The current credibility weight is transferred to a local variable
  credibility_weight -> W;

  history_of_incs_decs -> temp_list;

  ;;; Variables used in calculations of p and q
  0 -> P_positive;
  0 -> P_negative;
  0 -> Q_positive;
  0 -> Q_negative;

  until temp_list = []
  do
    hd(temp_list) -> i;

    ;;; These values are used to calculate p
    if i = 1
    then
      P_positive + 1.0 -> P_positive
    else
      P_negative + 1.0 -> P_negative
    endif;

    ;;; These values are used to calculate q
    ro/kappa -> epsilon;
    ro - epsilon -> ro;
```

```
  clean(ro) -> ro;

  if i = 1
  then
    Q_positive+ro -> Q_positive
  else
    Q_negative+ro -> Q_negative
  endif;

  tl(temp_list) -> temp_list
enduntil;

1.0 -> ro;

;;; Do not allow devisors equal to zero
if P_positive = 0
then
  1.0 -> P_positive
endif;

if P_negative = 0
then
  1.0 -> P_negative
endif;

if Q_positive=0
then
  1.0 -> Q_positive
endif;

if Q_negative = 0
then
  1.0 -> Q_negative
endif;


;;; Calculate p and q, depending on whether it is an increment or a decrement
;;; to the credibility weight
if inc_or_dec =1
then
  P_positive/P_negative -> p;
  Q_positive/Q_negative -> q
else
  P_negative/P_positive -> p;
  Q_negative/Q_positive -> q
endif;

;;; Calculate the functions f and g
if inc_or_dec = 1
then
  min((p+1),(q+1)) / max((q+1),(p+1)) -> f;
else
  min(((1/p)+1),((1/q)+1)) / max(((1/q)+1),((1/p)+1)) -> f;
endif;

1/kappa -> g;

;;; Calculate the increment, or decrement, to the credibility weight
if W > 0.5
then
  (0.5 - (W - 0.5)) * f * g -> Inc
else
  (0.5 - (0.5 - W)) * f * g -> Inc;
endif;

;;; Implement the change in the credibility weight
;;; and record this change
;;;
;;; This history is stored in a list called history_of_incs_decs and
;;; each new increment/decrement is added to the FRONT of the list.
;;; The value 1 signifies an increment, 0 signifies a decrement.
if inc_or_dec =1
then
  W + Inc -> credibility_weight;
  [1] <> history_of_incs_decs -> history_of_incs_decs
else
  W - Inc -> credibility_weight;
  [0] <> history_of_incs_decs -> history_of_incs_decs;
  (-1)*Inc -> Inc
endif;

;;; Record certain values in global variables
;;;   for monitoring the sensitivity of change ...
p -> p_value;
  q -> q_value;
  f -> f_value;
  Inc -> inc_value;

enddefmethod;


;;; METHOD: 8) assign_histories(number);
;;; This method assigns past histories stored in files
defmethod assign_histories(number);
  lvars number;

     hist_of_decisions(number) -> history_of_decisions;
  hist_of_incs_decs(number) -> history_of_incs_decs;

enddefmethod;


;;; METHOD: 9) save_to_file(x);
;;; Saves credibility weight, history of decisions,
;;;   history of increments/decrements and to files
defmethod save_to_file(x);
  lvars x filed;

  credibility_weight -> cred_weight(x);
  history_of_decisions -> hist_of_decisions(x);

  syscreate("credwt",1,"line") -> filed;
  cred_weight -> datafile(filed);
  sysclose(filed);
  syscreate("histdec",1,"line") -> filed;
  hist_of_decisions -> datafile(filed);
  sysclose(filed);
  syscreate("histid",1,"line") -> filed;
  hist_of_incs_decs -> datafile(filed);
  sysclose(filed);

enddefmethod;


;;; METHOD: 10) retrieve_from_file(x);
;;; Retrieves credibility weight, history of decisions,
;;;   history of increments/decrements from files
defmethod retrieve_from_file(x);
  lvars x filed;

  sysopen("credwt",0,"line") -> filed;
  datafile(filed) -> cred_weight;
  sysclose(filed);
  sysopen("histdec",0,"line") -> filed;
  datafile(filed) -> hist_of_decisions;
  sysclose(filed);
  sysopen("histid",0,"line") -> filed;
  datafile(filed) -> hist_of_incs_decs;
  sysclose(filed);

  if x = 1
  then
    'Professor DK Bowen' -> matrix_name;
    cred_weight(1) -> credibility_weight;
    hist_of_decisions(1) -> history_of_decisions;
    hist_of_incs_decs(1) -> history_of_incs_decs;

    if type_of_structure = 'substrate only'
    then
      sysopen("lofr1s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);

    elseif type_of_structure = 'a single layer'
    then
      sysopen("lofr1s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif type_of_structure = 'multiple layers'
    then
      sysopen("lofr1s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    else
      sysopen("lofr1s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    endif;
```

```
elseif x = 2
then
 'Professor BK Tanner' -> matrix_name;
 cred_weight(2) -> credibility_weight;
 hist_of_decisions(2) -> history_of_decisions;
 hist_of_incs_decs(2) -> history_of_incs_decs;

 if type_of_structure = 'substrate only'
 then
  sysopen("lofr2s1",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);

 elseif type_of_structure = 'a single layer'
 then
  sysopen("lofr2s2",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("lofr2s3",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 else
  sysopen("lofr2s4",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 endif;
elseif x = 3
then
 'Doctor Neil Loxley' -> matrix_name;
 cred_weight(3) -> credibility_weight;
 hist_of_decisions(3) -> history_of_decisions;
 hist_of_incs_decs(3) -> history_of_incs_decs;

 if type_of_structure = 'substrate only'
 then
  sysopen("lofr3s1",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);

 elseif type_of_structure = 'a single layer'
 then
  sysopen("lofr3s2",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("lofr3s3",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 else
  sysopen("lofr3s4",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 endif;

elseif x = 4
then
 'Doctor CR Thomas' -> matrix_name;
 cred_weight(4) -> credibility_weight;
 hist_of_decisions(4) -> history_of_decisions;
 hist_of_incs_decs(4) -> history_of_incs_decs;

 if type_of_structure = 'substrate only'
 then
  sysopen("lofr4s1",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);

 elseif type_of_structure = 'a single layer'
 then
  sysopen("lofr4s2",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("lofr4s3",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 else
  sysopen("lofr4s4",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 endif;

else
then
 'Dummy Expert' -> matrix_name;
 cred_weight(5) -> credibility_weight;
 hist_of_decisions(5) -> history_of_decisions;
 hist_of_incs_decs(5) -> history_of_incs_decs;

 if type_of_structure = 'substrate only'
 then
  sysopen("lofr5s1",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);

 elseif type_of_structure = 'a single layer'
 then
  sysopen("lofr5s2",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("lofr5s3",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 else
  sysopen("lofr5s4",0,"line") -> filed;
  datafile(filed) -> list_of_rules;
  sysclose(filed);
 endif;
endif;

enddefmethod;


;;; METHOD: 11) store_angst;
defmethod store_angst;
 lvars filed;

 if type_of_structure = 'substrate only'
 then
  syscreate("angst1",1,"line") -> filed;
  angst -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  syscreate("angst2",1,"line") -> filed;
  angst -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  syscreate("angst3",1,"line") -> filed;
  angst -> datafile(filed);
  sysclose(filed);
 else
  syscreate("angst4",1,"line") -> filed;
  angst -> datafile(filed);
  sysclose(filed);
 endif;

enddefmethod;

;;; METHOD: 12) printself;
defmethod printself;
 nl(2);pr('_____');nl(2);
 nl(2);pr('CONNECTION MATRIX CREATED FOR AN EXPERT');nl(2);
 pr('Name of the expert: ');pr(matrix_name);nl(2);
 pr('    List of rules used by this expert:');nl(1);
 pr('        ');pr(list_of_rules);nl(2);
 pr('    Credibility Weight for this expert = ');pr(credibility_weight);nl(2);
 pr('    History of most recent decisions using the rules of this expert');nl(1);
 pr('    1 is a successful decision, 2 is unsuccessful');nl(1);
 pr('        ');pr(history_of_decisions);nl(2);
 pr('_____');nl(2);
enddefmethod;

endflavour;

;;; _____
```

```
;;; 32. COMBINED_MATRIX OBJECT

flavour COMBINED_MATRIX;
 ivars matrix_name matrix;

 ;;; LIST OF METHODS
 ;;;
 ;;; 1) create(expert);
 ;;; 2) access_combined_matrix(i,j);
 ;;; 3) make_deductions();
 ;;; 4) invoke_unasserted_rules();
 ;;; 5) save_to_file();
 ;;; 6) retrieve_from_file();
 ;;; 7) printself(number_of_experts);


 ;;; METHOD: 1) create(expert);
 ;;; Create a combined connection matrix for a specified number of experts
 defmethod create(expert);
  lvars expert size_of_matrix i j k filed;

   'COMBINED CONNECTION MATRIX' -> matrix_name;

   number_of_rules * 2 -> size_of_matrix;
   newarray([1 ^size_of_matrix 1 ^size_of_matrix],0) -> matrix;

       nl(2);pr('Scalar multiply each connection matrix by the credibility
             weight of the expert.');nl(2);

   ;;; SCALAR MULTIPLY EACH CONNECTION MATRIX
   1 -> i;
   until i > number_of_experts
    do
    expert(i) <- scalar_multiply;
    i+1 -> i
   enduntil;

          pr('Use simple matrix addition to add the connection
                matrices together.');nl(2);

   1 -> i;
   1 -> j;
   1 -> k;
   until i > size_of_matrix
   do
    until j > size_of_matrix
    do
     until k > number_of_experts
     do
      matrix(i,j) + expert(k)<-access_cred_matrix(i,j) -> matrix(i,j);
      k+1 -> k
     enduntil;
     1 -> k;
     j+1 -> j
    enduntil;

    1 -> j;
    i+1 -> i

   enduntil;

enddefmethod;


;;; METHOD: 2) access_combined_matrix(i,j);
;;; This method accesses the value at specified indices of the combined matrix
defmethod access_combined_matrix(i,j);
 lvars i j;

 matrix(i,j)
enddefmethod;


;;; METHOD: 3) make_deductions;
;;; Use the combined connection matrix to choose those rules with the
;;; highest combined weightings and then fire this set of rules
defmethod make_deductions;
 lvars list_of_rules i j size_of_matrix largest_value;
 lvars rule_number no_of_rules rule_set mask;
 lvars arule premise consequence temp_list;
 lvars filed temp fired k;

 ;;; CREATE AN EMPTY LIST OF RULES
 [] -> list_of_rules;
 number_of_rules * 2 -> size_of_matrix;

 ;;; MAKE LIST OF RULES WITH THE HIGHEST VALUE IN THE
       COMBINED MATRIX
 1 -> i;
 1 -> j;

 until i > size_of_matrix
 do
  0 -> largest_value;
  0 -> rule_number;

  until j > size_of_matrix
  do
   if matrix(i,j) > largest_value
   then
    matrix(i,j) -> largest_value;
    j/2 -> rule_number
   endif;
   j+1 -> j
  enduntil;

  if rule_number > 0
  then
   ;;; Add rule number to the list of rules
   list_of_rules <> [^(rule_number)] -> list_of_rules
  endif;

  1 -> j;
  i+1 -> i
 enduntil;

 ;;; The length of this list equals the number of rules
 ;;; used by the expert system for the current decision
 length(list_of_rules) -> no_of_rules;
 list_of_rules -> temp_list;

 ;;; CREATE THE RULE SET (AS AN ARRAY OF RULES)
 newarray([1 3 1 ^no_of_rules]) -> rule_set;

 newarray([1 ^no_of_rules]) -> mask;
 if type_of_structure = 'substrate only'
 then
  sysopen("rmask1",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  sysopen("rmask2",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("rmask3",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
 else
  sysopen("rmask4",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
 endif;

 1 -> i;
 until i > no_of_rules
 do
  hd(temp_list) -> j;

  ;;; Add a rule to the ruleset
  rule(1,j) -> rule_set(1,i);
  rule(2,j) -> rule_set(2,i);
  rule(3,j) -> rule_set(3,i);

  masking_of_rule(j) -> mask(i);

  tl(temp_list) -> temp_list;
  i+1 -> i
 enduntil;

 list_of_rules -> original_list_of_rules;
 [] -> temp;
 1 -> k;

 nl(2);pr('Fire rules layer by layer and calculate the');nl(1);
 pr(' values of the conclusion variables.');nl(2);
```

```
  until k > 2
  do
   list_of_rules -> temp_list;

   1 -> i;
   until  i > no_of_rules
   do
    hd(temp_list) -> j;

    if mask(i) = "ASSERT" and hd(rule_set(3,i)) = k
    then

     ;;; create a rule object and use the rule
     make_instance([RULE]) ->arule;

     rule_set(1,i) -> premise;
     rule_set(2,i) -> consequence;

     arule <- use_rule(premise, consequence) -> fired;

     if fired=1
     then
      temp <> [^j] -> temp
     endif;
    endif;

    tl(temp_list) -> temp_list;
    i+1 -> i
   enduntil;

   k+1 -> k;
   load fuzzout.p;
  enduntil;

  nl(3);pr('The following rules are used in this decision: ');nl(2);
  pr(temp);nl(3);
  nl(3);
  pr('_____________________________________________________');nl(3);
  nl(3);pr('     DEDUCTIONS MADE BY THE FUZZY SYSTEM');nl(3);

  temp -> list_of_rules;

  return(list_of_rules);

enddefmethod;


;;;  METHOD:  4)  invoke_unasserted_rules;
;;;  Whenever there is not enough information produced by the ASSERTed
rules, then the
;;;  UNASSERTed rules are invoked. In this case, the user is prompted to
evaluate each
;;;  deduction as it appears on the screen.
defmethod invoke_unasserted_rules;
 lvars no_of_rules temp_list mask rule_set i j arule premise consequence;
 lvars answer temp fired rule_history filed;

 if type_of_structure = 'substrate only'
 then
  sysopen("rhist1",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  sysopen("rhist2",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("rhist3",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
 else
  sysopen("rhist4",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
 endif;

 ;;; The length of this list equals the number of rules
 ;;; used by the expert system for the current decision
 length(original_list_of_rules) -> no_of_rules;
 original_list_of_rules -> temp_list;

 ;;; CREATE THE RULE SET (AS AN ARRAY OF RULES)

 newarray([1 2 1 ^no_of_rules]) -> rule_set;
 newarray([1 ^no_of_rules]) -> mask;

 1 -> i;
 until i > no_of_rules
 do
  hd(temp_list) -> j;

  ;;; Add a rule to the ruleset
  rule(1,j) -> rule_set(1,i);
  rule(2,j) -> rule_set(2,i);

  masking_of_rule(j) -> mask(i);

  tl(temp_list) -> temp_list;

  i+1 -> i

 enduntil;

 ;;; USE THIS SET OF RULES TO MAKE DEDUCTIONS

 newarray([1 ^no_of_rules]) -> arule;

 original_list_of_rules -> temp_list;
 [] -> temp;

 1 -> i;
 until  i > no_of_rules
 do

  hd(temp_list) -> j;

  if mask(i) = "UNASSERT"
  then

   nl(5);pr('      DEDUCTION MADE BY AN
                UNASSERTED RULE');nl(3);

   make_instance([RULE]) ->arule(i);

   rule_set(1,i) -> premise;
   rule_set(2,i) -> consequence;

   ;;; create a rule object and use the rule
   arule(i) <- use_rule(premise, consequence) -> fired;

   if fired=1
   then
    temp <> [^j] -> temp
   endif;

   [] -> answer;
   until answer=[y] or answer=[Y] or answer=[n] or answer=[N]
   do
    nl(2);pr('  Is this deduction correct (Y/N)?');
    nl(1);pr('  Please type in your answer ...');

    readline() -> answer;

   enduntil;

   hd(temp_list) -> j;

   if length(rule_history(j)) > (N - 1)
   then
    tl(rule_history(j)) -> rule_history(i)
   endif;

   if answer=[y] or answer=[Y]
   then
    rule_history(j) <> [1]
   else
    rule_history(j) <> [0]
   endif;

  endif;

  i+1 -> i;
  tl(temp_list) -> temp_list

 enduntil;

 if type_of_structure = 'substrate only'
```

```
    then
      syscreate("rhist1",1,"line") -> filed;
      rule_history -> datafile(filed);
      sysclose(filed);
    elseif type_of_structure = 'a single layer'
    then
      syscreate("rhist2",1,"line") -> filed;
      rule_history -> datafile(filed);
      sysclose(filed);
    elseif type_of_structure = 'multiple layers'
    then
      syscreate("rhist3",1,"line") -> filed;
      rule_history -> datafile(filed);
      sysclose(filed);
    else
      syscreate("rhist4",1,"line") -> filed;
      rule_history -> datafile(filed);
      sysclose(filed);
    endif;

  enddefmethod;


  ;;; METHOD: 5) save_to_file();
  ;;; Saves the combined connection matrix to file
  defmethod save_to_file;
   lvars filed;

     syscreate("cmname",1,"line") -> filed;
     matrix_name -> datafile(filed);
     sysclose(filed);
     if type_of_structure = 'substrate only'
     then
       syscreate("cm1",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif type_of_structure = 'a single layer'
     then
       syscreate("cm2",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif type_of_structure = 'multiple layers'
     then
       syscreate("cm3",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     else
       syscreate("cm4",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     endif;
  enddefmethod;



  ;;; METHOD: 6) retrieve_from_file();
  ;;; Retrieves the combined connection matrix from file
  defmethod retrieve_from_file;
   lvars filed;

     sysopen("cmname",0,"line") -> filed;
     datafile(filed) -> matrix_name;
     sysclose(filed);
     sysopen("noexp",0,"line") -> filed;
     datafile(filed) -> number_of_experts;
     sysclose(filed);

     if type_of_structure = 'substrate only'
     then
       sysopen("cm1",0,"line") -> filed;
       datafile(filed) -> matrix;
       sysclose(filed);
       sysopen("rno1",0,"line") -> filed;
       datafile(filed) -> number_of_rules;
       sysclose(filed);
     elseif type_of_structure = 'a single layer'
     then
       sysopen("cm2",0,"line") -> filed;
       datafile(filed) -> matrix;
       sysclose(filed);
       sysopen("rno2",0,"line") -> filed;
       datafile(filed) -> number_of_rules;
       sysclose(filed);
     elseif type_of_structure = 'multiple layers'
     then
       sysopen("cm3",0,"line") -> filed;
       datafile(filed) -> matrix;
       sysclose(filed);
       sysopen("rno3",0,"line") -> filed;
       datafile(filed) -> number_of_rules;
       sysclose(filed);
     else
       sysopen("cm4",0,"line") -> filed;
       datafile(filed) -> matrix;
       sysclose(filed);
       sysopen("rno4",0,"line") -> filed;
       datafile(filed) -> number_of_rules;
       sysclose(filed);
     endif;
  enddefmethod;

  ;;; METHOD: 7) printself(number_of_experts);
  defmethod printself(number_of_experts);
   lvars number_of_experts i;

   nl(2);pr('_____');nl(2);
   nl(2);pr('COMBINED CONNECTION MATRIX CREATED FOR THE
             FOLLOWING EXPERTS');nl(2);

   1 -> i;
   until i > number_of_experts
   do
;;;      pr('Name of expert');pr(i);pr(':  ');pr(expert(i)<-matrix_name);nl(1);
     i+1 -> i
   enduntil;

   pr('_____');nl(2);
  enddefmethod;

endflavour;

;;; ____________________________________________________


;;; 33. FUNCTION to check if element is a member of a list

define ismember(element,list);
  if list matches [== ^element ==]
    then true
    else false
  endif
enddefine;


define get_median_value(list);
  lvars list list2 check a median_value;

  ;;; RANK THE LIST
  0 -> check;
  while check=0
  do
    1 -> check;
    [] -> list2;
    hd(list) -> a;
    list2 <> [^a] -> list2;
    tl(list) -> list;
    until list = []
    do
      if hd(list) < a
      then
        hd(list) -> a;
        [^a] <> list2 -> list2;
        0 -> check
      else
        hd(list) -> a;
        list2 <> [^a] -> list2;
      endif;
      tl(list) -> list
    enduntil;
    list2 -> list;
  endwhile;

  list(9) -> median_value;
  return(median_value);
enddefine;
```

```
;;; 34. OBJECT USED TO CONTROL THE MECHANISMS OF THE
;;;     EXPERT SYSTEM
;;;
;;; FUZZY_SYSTEM OBJECT
;;;
;;;
;;; LIST OF METHODS:
;;;
;;;    1) initialise_rocking_curve_parameters(x);
;;;    2) store_rocking_curve_parameters(x);
;;;    3) create_the_connection_matrices;
;;;    4) load_the_rules_from_file;
;;;    5) calculate_the_outcome() -> decision;
;;;    6) use_the_unasserted_rules;
;;;    7) update_the_histories;
;;;    8) retrieve_from_files_1();
;;;    9) save_to_files_1();
;;;   10) check_experts_credibility_weights(an_expert);
;;;   11) monitor_parameters_for_sensitivity(i);
;;;   12) set_up_some_variables;
;;;   13) check_ASSERT_UNASSERT_functions();
;;;   14) save_to_files_2();
;;;   15) fine_tuning_membership_functions;
;;;   16) retrieve_from_files_2();
;;;   17) choose_which_variable(number1, number2);
;;;   18) save_to_files_3();
;;;   19) were_changes_made_to_credibility_weights();
;;;   20) record_variable_values();
;;;   21) run_the_system();

flavour FUZZY_SYSTEM;
 ivars outcome_of_decision;


 ;;; METHOD 1: initialise_rocking_curve_parameters(x);
 ;;; This method gives initial (default) values to variables that
 ;;; store the rocking curve parameters

 defmethod initialise_rocking_curve_parameters(x);
  lvars x;

  if x=1
  then
   ;;; From Experiment Frame 1
   0 -> wavelength;
   0 -> h_reflection_index;
   0 -> k_reflection_index;
   0 -> l_reflection_index;
   "VOID" -> substrate_material;
   "VOID" -> reflection_orientation;
   0 -> h_surface_normal_index;
   0 -> k_surface_normal_index;
   0 -> l_surface_normal_index;
   0 -> Bragg_angle;
   0 -> start_of_scan_range;
   0 -> end_of_scan_range;
  endif;

  if x = 1
  then
   ;;; From Layer-Frame
   "VOID" -> layer_material;
   "VOID" -> layer_chemistry;
   0 -> layer_thickness;
   0 -> layer_thickness_greater_than_half_micron;
   0 -> layer_thickness_less_than_5_microns;
  endif;

  ;;; From Substrate-Peak Frame
  0 -> substrate_FWHM;
  999 -> substrate_peak_broadening;
  0 -> substrate_integrated_intensity_of_peak;
  999 -> substrate_asymmetry_in_peak;
  0 -> split_substrate_peak;

  ;;; From Layer-Peak Frame
  0 -> layer_FWHM;
  "VOID" -> layer_peak_broadening;
  0 -> layer_integrated_intensity_of_peak;
  0 -> layer_integrated_intensity_of_peak_is_zero;
  999 -> layer_asymmetry_in_peak;
  999 -> layer_wedge_shaped_peak;
  0 -> layer_split_peak;
  999 -> intensity_of_layer_peak;

  ;;; From Multiple-Layer-Peaks Frame
  "VOID" -> multiple_layers_peak_broadening;
  "VOID" -> multiple_layers_any_asymmetry;
  999 -> multiple_layers_asymmetry_in_peak;
  "VOID" -> multiple_layers_any_wedge_shaped_peaks;
  999 -> multiple_layers_wedge_shaped_peak;

  ;;; From Single-Satellite-Peak Frame
  0 -> satellite_intensity_of_peak;
  0 -> satellite_FWHM;
  0 -> satellite_order;
  0 -> satellite_position;

  ;;; From Satellite-Peaks Frame
  999 -> satellite_visibility;
  0 -> satellite_spacing_of_peaks;
  999 -> satellite_asymmetry_of_plus_and_minus_peaks;
  0 -> satellite_number_of_peaks;
  0 -> satellite_relative_intensities;
  0 -> satellite_relative_width_of_peaks;
  0 -> satellite_relative_integrated_intensities;
  "VOID" -> satellite_peak_splitting;
  0 -> satellite_order_1;
  0 -> satellite_position_1;
  0 -> satellite_order_2;
  0 -> satellite_position_2;
  1 -> satellite_spacing_greater_than_zero;
  0 -> satellite_subsidiary_interference_effects;
  0 -> satellite_broadening_of_higher_order_peaks;
  0 -> satellite_relative_width_of_peaks_greater_than_zero;
  0 -> satellite_relative_integrated_intensities_greater_than_zero;
  0 -> thickness_of_superlattice_less_than_half_micron;

  ;;; From Zero-Order Peak Frame
  0 -> zero_order_FWHM;
  0 -> zero_order_integrated_intensity_of_peak;
  999 -> zero_order_asymmetry_in_peak;

  ;;; From Interference-Fringes_frame
  999 -> interference_fringes;
  0 -> spacing_of_interference_fringes;
  0 -> spacing_of_interference_fringes_is_low;
  999 -> visibility_of_interference_fringes;

  ;;; From Substrate-Only Frame
  if x=1
  then
   1 -> number_of_layers;
  endif;
  0 -> number_of_peaks_is_one;
  0 -> number_of_peaks_is_more_than_one;
  0 -> number_of_peaks_is_none;

  ;;; From Single-Layer-Frame
  0 -> number_of_peaks_is_two;
  0 -> number_of_peaks_is_more_than_two;
  0 -> substrate_layer_peak_overlap;
  0 -> substrate_material_equal_to_layer;
  0 -> peak_splitting;
  0 -> peak_separation_is_low;
  0 -> peak_splitting_is_zero;
  0 -> peak_splitting_less_than_three_times_width_of_peak;
  0 -> peak_splitting_is_high;
  if x=1
  then
   "VOID" -> relaxed_layer;
  endif;
  0 -> three_times_width_of_peak;

  ;;; From A-Number-of-Layers-Frame
  if x=1
  then
   "VOID" -> layer_composition;
   0 -> number_of_blocks
  endif;

  ;;; From Superlattice-Frame
  if x=1
  then
   0 -> thickness_of_superlattice;
  endif;
  "VOID" -> more_than_two_peaks;
  "VOID" -> identify_the_peaks;
```

```
;;; From Multiple-Layers Frame
if x=1
then
  "VOID" -> layers_in_blocks;
endif;
0 -> correspondence_between_layers_and_peaks;

;;; From Superlattice-With-a-Few_Layers Frame
if x=1
then
  "VOID" -> top_layer_composition;
  0 -> top_layer_thickness;
  "VOID" -> bottom_layer_composition;
  0 -> bottom_layer_thickness;
  0 -> number_of_top_layers;
  0 -> number_of_bottom_layers;
endif;
0 -> number_of_peaks;
"VOID" -> overlap_of_peaks;

if x=1
then
  0 -> experimental_mismatch;
  1 -> relaxed_mismatch;
  0 -> relaxed_mismatch_is_high;
  1 -> spacing_of_planes;
  0 -> lattice_parameter;
  0 -> period_of_superlattice;
  0 -> period_dispersion;
  0 -> thickness_of_layer
endif;

enddefmethod;


;;; METHOD 2: store_rocking_curve_parameters(x);
;;; This method stores rocking curve parameters to file

defmethod store_rocking_curve_parameters(x);
lvars filed substrate layer ratio_of_integrated_intensities;

 if type_of_structure = 'substrate only'
 then
  newarray([1 25]) ->dummy_array;

  type_of_structure -> dummy_array(1);
  wavelength -> dummy_array(2);
  h_reflection_index -> dummy_array(3);
  k_reflection_index -> dummy_array(4);
  l_reflection_index -> dummy_array(5);
  substrate_material -> dummy_array(6);
  reflection_orientation -> dummy_array(7);
  h_surface_normal_index -> dummy_array(8);
  k_surface_normal_index -> dummy_array(9);
  l_surface_normal_index -> dummy_array(10);
  Bragg_angle -> dummy_array(11);
  start_of_scan_range -> dummy_array(12);
  end_of_scan_range -> dummy_array(13);
  substrate_FWHM -> dummy_array(14);
  substrate_peak_broadening -> dummy_array(15);
  substrate_integrated_intensity_of_peak -> dummy_array(16);
  substrate_asymmetry_in_peak -> dummy_array(17);
  interference_fringes -> dummy_array(18);
  spacing_of_interference_fringes -> dummy_array(19);
  visibility_of_interference_fringes -> dummy_array(20);
  number_of_peaks -> dummy_array(21);
  type_of_structure_is_substrate_only -> dummy_array(22);
  number_of_peaks_is_one -> dummy_array(23);
  number_of_peaks_is_more_than_one -> dummy_array(24);
  number_of_peaks_is_none -> dummy_array(25);

 elseif type_of_structure = 'a single layer'
 then
  newarray([1 47]) -> dummy_array;

  type_of_structure -> dummy_array(1);
  wavelength -> dummy_array(2);
  h_reflection_index -> dummy_array(3);
  k_reflection_index -> dummy_array(4);
  l_reflection_index -> dummy_array(5);
  substrate_material -> dummy_array(6);
  reflection_orientation -> dummy_array(7);
  h_surface_normal_index -> dummy_array(8);
  k_surface_normal_index -> dummy_array(9);
  l_surface_normal_index -> dummy_array(10);
  Bragg_angle -> dummy_array(11);
  start_of_scan_range -> dummy_array(12);
  end_of_scan_range -> dummy_array(13);
  layer_material -> dummy_array(14);
  layer_chemistry -> dummy_array(15);
  layer_thickness -> dummy_array(16);
  substrate_FWHM -> dummy_array(17);
  substrate_peak_broadening -> dummy_array(18);
  substrate_integrated_intensity_of_peak -> dummy_array(19);
  substrate_asymmetry_in_peak -> dummy_array(20);
  layer_FWHM -> dummy_array(21);
  layer_peak_broadening -> dummy_array(22);
  layer_integrated_intensity_of_peak -> dummy_array(23);
  layer_asymmetry_in_peak -> dummy_array(24);
  layer_wedge_shaped_peak -> dummy_array(25);
  if layer_split_peak = [Y]
  then
    1 -> layer_split_peak;
  else
    0 -> layer_split_peak;
  endif;
  layer_split_peak -> dummy_array(26);
  interference_fringes -> dummy_array(27);
  spacing_of_interference_fringes -> dummy_array(28);
  visibility_of_interference_fringes -> dummy_array(29);
  number_of_peaks -> dummy_array(30);
  type_of_structure_is_single_layer -> dummy_array(31);
  number_of_peaks_is_one -> dummy_array(32);
  number_of_peaks_is_two -> dummy_array(33);
  number_of_peaks_is_more_than_two -> dummy_array(34);
  number_of_peaks_is_none -> dummy_array(35);
  substrate_material_equal_to_layer -> dummy_array(36);
  peak_separation_is_low -> dummy_array(37);
  layer_integrated_intensity_of_peak_is_zero -> dummy_array(38);
  layer_thickness_greater_than_half_micron -> dummy_array(39);
  layer_thickness_less_than_5_microns -> dummy_array(40);
  peak_splitting_is_zero -> dummy_array(41);
  peak_splitting_less_than_three_times_width_of_peak->dummy_array(42);
  relaxed_mismatch_is_high -> dummy_array(43);
  peak_splitting_is_high -> dummy_array(44);
  spacing_of_interference_fringes_is_low -> dummy_array(45);

  if x = 1
  then
    substrate_FWHM * substrate_integrated_intensity_of_peak -> substrate;
    layer_FWHM * layer_integrated_intensity_of_peak -> layer;
    layer / substrate -> ratio_of_integrated_intensities;
    ratio_of_integrated_intensities -> dummy_array(46);
  else
    layer_integrated_intensity_of_peak/substrate_integrated_intensity_of_peak
          -> ratio_of_integrated_intensities;
    ratio_of_integrated_intensities -> dummy_array(46);
  endif;
  intensity_of_layer_peak -> dummy_array(47);

elseif type_of_structure = 'multiple layers'
then
  newarray([1 33]) ->dummy_array;

  type_of_structure -> dummy_array(1);
  wavelength -> dummy_array(2);
  h_reflection_index -> dummy_array(3);
  k_reflection_index -> dummy_array(4);
  l_reflection_index -> dummy_array(5);
  substrate_material -> dummy_array(6);
  reflection_orientation -> dummy_array(7);
  h_surface_normal_index -> dummy_array(8);
  k_surface_normal_index -> dummy_array(9);
  l_surface_normal_index -> dummy_array(10);
  Bragg_angle -> dummy_array(11);
  start_of_scan_range -> dummy_array(12);
  end_of_scan_range -> dummy_array(13);
  substrate_FWHM -> dummy_array(14);
  substrate_peak_broadening -> dummy_array(15);
  substrate_integrated_intensity_of_peak -> dummy_array(16);
  substrate_asymmetry_in_peak -> dummy_array(17);
  multiple_layers_peak_broadening -> dummy_array(18);
  multiple_layers_any_asymmetry -> dummy_array(19);
  multiple_layers_asymmetry_in_peak -> dummy_array(20);
  multiple_layers_any_wedge_shaped_peaks -> dummy_array(21);
  multiple_layers_wedge_shaped_peak -> dummy_array(22);
  layers_in_blocks -> dummy_array(23);
  number_of_layers -> dummy_array(24);
```

```
  number_of_blocks -> dummy_array(25);
  number_of_peaks -> dummy_array(26);
  interference_fringes -> dummy_array(27);
  spacing_of_interference_fringes -> dummy_array(28);
  visibility_of_interference_fringes -> dummy_array(29);
  type_of_structure_is_multiple_layers -> dummy_array(30);
  correspondence_between_layers_and_peaks -> dummy_array(31);
  number_of_peaks_is_none -> dummy_array(32);
  split_substrate_peak -> dummy_array(33);

elseif type_of_structure = 'MQW structure'
then
  newarray([1 60]) ->dummy_array;

  type_of_structure -> dummy_array(1);
  wavelength -> dummy_array(2);
  h_reflection_index -> dummy_array(3);
  k_reflection_index -> dummy_array(4);
  l_reflection_index -> dummy_array(5);
  substrate_material -> dummy_array(6);
  reflection_orientation -> dummy_array(7);
  h_surface_normal_index -> dummy_array(8);
  k_surface_normal_index -> dummy_array(9);
  l_surface_normal_index -> dummy_array(10);
  Bragg_angle -> dummy_array(11);
  start_of_scan_range -> dummy_array(12);
  end_of_scan_range -> dummy_array(13);
  satellite_visibility -> dummy_array(14);
  satellite_spacing_of_peaks -> dummy_array(15);
  satellite_asymmetry_of_plus_and_minus_peaks -> dummy_array(16);
  satellite_number_of_peaks -> dummy_array(17);
  satellite_relative_intensities -> dummy_array(18);
  satellite_relative_width_of_peaks -> dummy_array(19);
  satellite_relative_integrated_intensities -> dummy_array(20);
  satellite_subsidiary_interference_effects -> dummy_array(21);
  satellite_peak_splitting -> dummy_array(22);
  satellite_broadening_of_higher_order_peaks -> dummy_array(23);
  substrate_FWHM -> dummy_array(24);
  substrate_peak_broadening -> dummy_array(25);
  substrate_integrated_intensity_of_peak -> dummy_array(26);
  substrate_asymmetry_in_peak -> dummy_array(27);
  zero_order_FWHM -> dummy_array(28);
  zero_order_integrated_intensity_of_peak -> dummy_array(29);
  zero_order_asymmetry_in_peak -> dummy_array(30);
  interference_fringes -> dummy_array(31);
  spacing_of_interference_fringes -> dummy_array(32);
  visibility_of_interference_fringes -> dummy_array(33);
  number_of_layers -> dummy_array(34);
  0 -> layer_composition;
  layer_composition -> dummy_array(35);
  layer_thickness -> dummy_array(36);
  number_of_blocks -> dummy_array(37);
  thickness_of_superlattice -> dummy_array(38);
  more_than_two_peaks -> dummy_array(39);
  identify_the_peaks -> dummy_array(40);
  type_of_structure_is_MQW -> dummy_array(41);
  satellite_spacing_greater_than_zero -> dummy_array(42);
  satellite_relative_width_of_peaks_greater_than_zero -> dummy_array(43);
  satellite_relative_integrated_intensities_greater_than_zero
          -> dummy_array(44);
  thickness_of_superlattice_less_than_half_micron -> dummy_array(45);
  number_of_peaks_is_none -> dummy_array(46);

  0 -> top_layer_composition;
  top_layer_composition -> dummy_array(47);
  0 -> top_layer_thickness;
  top_layer_thickness -> dummy_array(48);
  bottom_layer_composition -> dummy_array(49);
  bottom_layer_thickness -> dummy_array(50);
  number_of_top_layers -> dummy_array(51);
  number_of_bottom_layers -> dummy_array(52);
  number_of_peaks -> dummy_array(53);
  overlap_of_peaks -> dummy_array(54);
  multiple_layers_peak_broadening -> dummy_array(55);
  multiple_layers_any_asymmetry -> dummy_array(56);
  multiple_layers_asymmetry_in_peak -> dummy_array(57);
  multiple_layers_any_wedge_shaped_peaks -> dummy_array(58);
  multiple_layers_wedge_shaped_peak -> dummy_array(59);
  type_of_structure_has_additional_layers -> dummy_array(60);

endif;

if x=1
then
  syscreate("exper",1,"line") -> filed;
  dummy_array -> datafile(filed);
  sysclose(filed);
else
  syscreate("simul",1,"line") -> filed;
  dummy_array -> datafile(filed);
  sysclose(filed);
endif;

enddefmethod;


;;; METHOD 3: create_the_connection_matrices;
;;; This method creates the connection matrices for three experts

defmethod create_the_connection_matrices;
 lvars an_expert ruleset1 ruleset2 ruleset3 ruleset4 filed;

 sysopen("noexp",0,"line") -> filed;
 datafile(filed) -> number_of_experts;
 sysclose(filed);

 ;;; This array will contain all the connection matrices for all experts
 newarray([1 ^number_of_experts]) -> an_expert;

 ;;; Basic connection matrices were created by the
 ;;; program 'initial.p' and stored in a series of files
 if type_of_structure = 'substrate only'
 then
  sysopen("rno1",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
  sysopen("mc1s1",0,"line") -> filed;
  datafile(filed) -> ruleset1;
  sysclose(filed);
  sysopen("mc2s1",0,"line") -> filed;
  datafile(filed) -> ruleset2;
  sysclose(filed);
  sysopen("mc3s1",0,"line") -> filed;
  datafile(filed) -> ruleset3;
  sysclose(filed);
  sysopen("mc4s1",0,"line") -> filed;
  datafile(filed) -> ruleset4;
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  sysopen("rno2",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
   sysopen("mc1s2",0,"line") -> filed;
  datafile(filed) -> ruleset1;
  sysclose(filed);
  sysopen("mc2s2",0,"line") -> filed;
  datafile(filed) -> ruleset2;
  sysclose(filed);
  sysopen("mc3s2",0,"line") -> filed;
  datafile(filed) -> ruleset3;
  sysclose(filed);
  sysopen("mc4s2",0,"line") -> filed;
  datafile(filed) -> ruleset4;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("rno3",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
   sysopen("mc1s3",0,"line") -> filed;
  datafile(filed) -> ruleset1;
  sysclose(filed);
  sysopen("mc2s3",0,"line") -> filed;
  datafile(filed) -> ruleset2;
  sysclose(filed);
  sysopen("mc3s3",0,"line") -> filed;
  datafile(filed) -> ruleset3;
  sysclose(filed);
  sysopen("mc4s3",0,"line") -> filed;
  datafile(filed) -> ruleset4;
  sysclose(filed);
 else
  sysopen("rno4",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
   sysopen("mc1s4",0,"line") -> filed;
  datafile(filed) -> ruleset1;
```

```
    sysclose(filed);
    sysopen("me2s4",0,"line") -> filed;
    datafile(filed) -> ruleset2;
    sysclose(filed);
    sysopen("me3s4",0,"line") -> filed;
    datafile(filed) -> ruleset3;
    sysclose(filed);
    sysopen("me4s4",0,"line") -> filed;
    datafile(filed) -> ruleset4;
    sysclose(filed);
  endif;

  sysopen("credwt",0,"line") -> filed;
  datafile(filed) -> cred_weight;
  sysclose(filed);

  make_instance([CONNECTION_MATRIX]) -> an_expert(1);
  nl(1);pr('Creating the connection matrix for DKB');
  an_expert(1) <- create_a_connection_matrix('Professor DK Bowen',
                  ruleset1);
  an_expert(1) <- assign_credibility_weight(cred_weight(1));

  make_instance([CONNECTION_MATRIX]) -> an_expert(2);
  nl(1);pr('Creating the connection matrix for BKT');
  an_expert(2) <- create_a_connection_matrix('Professor BK Tanner',
                  ruleset2);
  an_expert(2) <- assign_credibility_weight(cred_weight(2));

  make_instance([CONNECTION_MATRIX]) -> an_expert(3);
  nl(1);pr('Creating the connection matrix for NL');
  an_expert(3) <- create_a_connection_matrix('Doctor N Loxley', ruleset3);
  an_expert(3) <- assign_credibility_weight(cred_weight(3));

  make_instance([CONNECTION_MATRIX]) -> an_expert(4);
  nl(1);pr('Creating the connection matrix for DE');
  an_expert(4) <- create_a_connection_matrix('DUMMY expert', ruleset4);
  an_expert(4) <- assign_credibility_weight(cred_weight(4));

  return(an_expert)

enddefmethod;


;;; METHOD 4: load_the_rules_from_file;
;;; This method creates the connection matrices for three experts

defmethod load_the_rules_from_file;
 lvars filed;

 ;;; RULES ARE STORED IN A 2XM MATRIX (where M is the number
 ;;;  of rules)
 newarray([1 3 1 ^number_of_rules]) -> rule;

 nl(2);pr('Loading the ruleset from file');nl(2);
 ;;; Rules are stored in files "rules1.p", "rules2.p", "rules3.p", "rules4.p"
 if type_of_structure = 'substrate only'
 then
   sysopen("fuzzno1",0,"line") -> filed;
   datafile(filed) -> number_of_fuzzy_variables;
   sysclose(filed);
   sysopen("rvar1",0,"line") -> filed;
   datafile(filed) -> rule_variable;
   sysclose(filed);
   sysopen("memfunc1",0,"line") -> filed;
   datafile(filed) -> membership_functions;
   sysclose(filed);
   sysopen("memf1",0,"line") -> filed;
   datafile(filed) -> mf;
   sysclose(filed);
   load rules1.p;
 elseif type_of_structure = 'a single layer'
 then
   sysopen("fuzzno2",0,"line") -> filed;
   datafile(filed) -> number_of_fuzzy_variables;
   sysclose(filed);
   sysopen("rvar2",0,"line") -> filed;
   datafile(filed) -> rule_variable;
   sysclose(filed);
   sysopen("memfunc2",0,"line") -> filed;
   datafile(filed) -> membership_functions;
   sysclose(filed);
   sysopen("memf2",0,"line") -> filed;
   datafile(filed) -> mf;
   sysclose(filed);
   load rules2.p;
 elseif type_of_structure = 'multiple layers'
 then
   sysopen("fuzzno3",0,"line") -> filed;
   datafile(filed) -> number_of_fuzzy_variables;
   sysclose(filed);
   sysopen("rvar3",0,"line") -> filed;
   datafile(filed) -> rule_variable;
   sysclose(filed);
   sysopen("memfunc3",0,"line") -> filed;
   datafile(filed) -> membership_functions;
   sysclose(filed);
   sysopen("memf3",0,"line") -> filed;
   datafile(filed) -> mf;
   sysclose(filed);
   load rules3.p;
 else
   sysopen("fuzzno4",0,"line") -> filed;
   datafile(filed) -> number_of_fuzzy_variables;
   sysclose(filed);
   sysopen("rvar4",0,"line") -> filed;
   datafile(filed) -> rule_variable;
   sysclose(filed);
   sysopen("memfunc4",0,"line") -> filed;
   datafile(filed) -> membership_functions;
   sysclose(filed);
   sysopen("memf4",0,"line") -> filed;
   datafile(filed) -> mf;
   sysclose(filed);
   load rules4.p;
 endif;
 ;;; rules are stored in the array rule(3,m)

enddefmethod;


;;; METHOD 5: calculate_the_outcome() -> decision;
;;; This method calculates the outcome of the current decision

defmethod calculate_the_outcome -> decision;
 lvars filed experimental_curve simulated_curve variable_list;
 lvars number total_of_outcomes ratio_of_values i decision;
 lvars history_of_real_values;

 ;;; Load the descriptions of the curves from data files: exper & simul

 sysopen("exper",0,"line") -> filed;
 datafile(filed) -> experimental_curve;
 sysclose(filed);

 sysopen("simul",0,"line") -> filed;
 datafile(filed) -> simulated_curve;
 sysclose(filed);

 ;;; Choose the relevant variables for the current structure
 ;;; The numbers of the variables are stored in a list: variable_list
 if (experimental_curve(1)='substrate only')
 then
   [14 15 16 17 23 24 25] -> variable_list;
   ;;; Translate number_of_peaks into numerical values
 elseif (experimental_curve(1)='a single layer')
 then
   [17 18 20 21 22 24 25 26 32 33 34 35 42 44 46] -> variable_list;
 elseif experimental_curve(1)='multiple layers'
 then
   [14 15 16 17 18 19 20 21 22 31 32] -> variable_list;
 else
   if type_of_structure2 = 'additional layers'
   then
     [14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
       39 40 42 55 56 57 58 59] -> variable_list;
   else
     [14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
       39 40 42] -> variable_list;
   endif;
 endif;

 ;;; Calculate the outcome of the decision by objective means
 length(variable_list) -> number;
 0.0 -> ratio_of_values;
 0.0 -> total_of_outcomes;

 0 -> i;
 until variable_list=[]
```

```
  do
   hd(variable_list) -> i;

   ;;; Translate Y/N answers into Boolean values
   if experimental_curve(i)=[Y]
   then
    1.0 -> experimental_curve(i)
   elseif experimental_curve(i)=[N]
   then
    0.0 -> experimental_curve(i)
   else
   endif;

   if simulated_curve(i)=[Y]
   then
    1.0 -> simulated_curve(i)
   elseif simulated_curve(i)=[N]
   then
    0.0 -> simulated_curve(i)
   else
   endif;


   if experimental_curve(i)="VOID"
   then
    1.0 -> experimental_curve(i)
   endif;

   if simulated_curve(i)="VOID"
   then
    1.0 -> simulated_curve(i)
   endif;

   ;;; Calculate a ratio of experimental to simulated for each variable
   if experimental_curve(i)=0 and simulated_curve(i)=0
   then
    1.0 -> ratio_of_values;
   else
    min(experimental_curve(i), simulated_curve(i)) /
        max(experimental_curve(i), simulated_curve(i)) -> ratio_of_values;
   endif;

   total_of_outcomes+ratio_of_values -> total_of_outcomes;
   tl(variable_list) -> variable_list;
  enduntil;

  ;;; Calculate an overall outcome for this decision
  (total_of_outcomes / number) -> decision;

  ;;; Update history of decision values
  sysopen("histrv",0,"line") -> filed;
  datafile(filed) -> history_of_real_values;
  sysclose(filed);

  clean(decision) -> decision;
  history_of_real_values <> [^decision] -> history_of_real_values;

  syscreate("histrv",1,"line") -> filed;
  history_of_real_values -> datafile(filed);
  sysclose(filed);

enddefmethod;


;;; METHOD 6: use_the_unasserted_rules;
;;; This method uses unasserted rules when the decision is
;;;  close to but just below the threshold

defmethod use_the_unasserted_rules;
 lvars combined_connection_matrix;

 make_instance([COMBINED_MATRIX]) -> combined_connection_matrix;
 combined_connection_matrix <- retrieve_from_file;
 combined_connection_matrix <- invoke_unasserted_rules;
 cancel combined_connection_matrix;

enddefmethod;

;;; METHOD 7: update_the_histories;
;;; This method updates lists recording the histories of decisions
;;;  for experts, rules and variables

defmethod update_the_histories;
 lvars an_expert i temp_list;

 ;;; Retrieve the individual connection matrices from files
 newarray([1 ^number_of_experts]) -> an_expert;
 1 -> i;
 until i > number_of_experts
 do
  make_instance([CONNECTION_MATRIX]) -> an_expert(i);
  an_expert(i) <- retrieve_from_file(i);
  i + 1 -> i
 enduntil;

 ;;; METHOD 8)
 ^retrieve_from_files_1;

 ;;; Check through the list of rules used in this expert system decision.
 ;;; If an expert's rules are used then the number of this expert is
 ;;;  recorded in the list: list_of_experts
 [] -> list_of_experts;
 1 -> i;
 until i > number_of_experts
 do
  list_of_rules_used -> temp_list;

  until temp_list = []
  do
   if ismember(hd(temp_list), an_expert(i)<-list_of_rules)
   then
    list_of_experts <> [^i] -> list_of_experts;
    [] -> temp_list
   else
    tl(temp_list) -> temp_list;
   endif;
  enduntil;
  i+1 -> i
 enduntil;

 ;;; If the rules of an expert are used, then record the outcome for this
 ;;; decision is recorded in the history of decisions for this expert
 list_of_experts -> temp_list;
 until temp_list = []
 do
  hd(temp_list) -> i;
  an_expert(i) <- update_history(outcome_of_decision);
  tl(temp_list) -> temp_list
 enduntil;

 ;;; Check for each rule if it was used in the decision.
 ;;;  If yes, then record this in its rule history.
 ;;; New decision is recorded at the END of the list.
 list_of_rules_used -> temp_list;
 until temp_list = []
 do
  hd(temp_list) -> i;
  if masking_of_rule(i)="ASSERT"
  then
   if length(rule_history(i)) > (N - 1)
   then
    tl(rule_history(i)) -> rule_history(i)
   endif;
   rule_history(i) <> [^outcome_of_decision] -> rule_history(i)
  endif;
  tl(temp_list) -> temp_list
 enduntil;

 ;;; METHOD 9)
 ^save_to_files_1;

 return(an_expert)

enddefmethod;


;;; METHOD 8: retrieve_from_files_1();
defmethod retrieve_from_files_1;
 lvars filed;

 if type_of_structure = 'substrate only'
 then
  sysopen("rhist1",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  sysopen("rhist2",0,"line") -> filed;
  datafile(filed) -> rule_history;
```

```
    sysclose(filed);
    sysopen("rmask2",0,"line") -> filed;
    datafile(filed) -> masking_of_rule;
    sysclose(filed);
  elseif type_of_structure = 'multiple layers'
  then
    sysopen("rhist3",0,"line") -> filed;
    datafile(filed) -> rule_history;
    sysclose(filed);
  else
    sysopen("rhist4",0,"line") -> filed;
    datafile(filed) -> rule_history;
    sysclose(filed);
  endif;

enddefmethod;


;;; METHOD 9: save_to_files_1();
defmethod save_to_files_1;
  lvars filed;

  if type_of_structure = 'substrate_only'
  then
    syscreate("rhist1",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'a single layer'
  then
    syscreate("rhist2",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'multiple layers'
  then
    syscreate("rhist3",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
  else
    syscreate("rhist4",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
  endif;

enddefmethod;

;;; METHOD 10: check_experts_credibility_weights(an_expert);
defmethod check_experts_credibility_weights(an_expert);
  lvars an_expert i percent_for_success percent_for_failure;
  lvars temp_list SPECIAL_VARIABLE filed;

  ;;;  i indexes the experts
  ;;;  If one expert's rules are successful in percent_for_success percent
  ;;;  of expert system decisions then increase that expert's credibility weighting
  ;;;  and vice-versa if unsuccessful ... using percent_for_failure

  sysopen("pfors",0,"line") -> filed;
  datafile(filed) -> percent_for_success;
  sysclose(filed);
  sysopen("pforf",0,"line") -> filed;
  datafile(filed) -> percent_for_failure;
  sysclose(filed);

  list_of_experts -> temp_list;
  0 -> SPECIAL_VARIABLE;
  until temp_list = []
  do
    hd(temp_list) -> i;
    an_expert(i) <- check_success(percent_for_success, percent_for_failure)
                    -> SPECIAL_VARIABLE;
    an_expert(i) <- save_to_file(i);
    if SPECIAL_VARIABLE = 1
    then
      ;;;  METHOD 11)
      ^monitor_parameters_for_sensitivity(i);
      0 -> SPECIAL_VARIABLE
    endif;
    tl(temp_list) -> temp_list
  enduntil;

  ;;; Reassign global variables to null values
  0 -> cred_weight;
  0 -> hist_of_decisions;
  0 -> hist_of_incs_decs;
enddefmethod;

;;; METHOD 11: monitor_parameters_for_sensitivity(i);
defmethod monitor_parameters_for_sensitivity(i);
  lvars i filed;
  lvars p_values q_values f_values inc_values;

  sysopen("pfunc",0,"line") -> filed;
  datafile(filed) -> p_values;
  sysclose(filed);
  sysopen("qfunc",0,"line") -> filed;
  datafile(filed) -> q_values;
  sysclose(filed);
  sysopen("ffunc",0,"line") -> filed;
  datafile(filed) -> f_values;
  sysclose(filed);
  sysopen("incvals",0,"line") -> filed;
  datafile(filed) -> inc_values;
  sysclose(filed);

  p_values(i) <> [^p_value] -> p_values(i);
  q_values(i) <> [^q_value] -> q_values(i);
  f_values(i) <> [^f_value] -> f_values(i);
  inc_values(i) <> [^inc_value] -> inc_values(i);

  syscreate("pfunc",1,"line") -> filed;
  p_values -> datafile(filed);
  sysclose(filed);
  syscreate("qfunc",1,"line") -> filed;
  q_values -> datafile(filed);
  sysclose(filed);
  syscreate("ffunc",1,"line") -> filed;
  f_values -> datafile(filed);
  sysclose(filed);
  syscreate("incvals",1,"line") -> filed;
  inc_values -> datafile(filed);
  sysclose(filed);

enddefmethod;


;;; METHOD 12: set_up_some_variables;
defmethod set_up_some_variables;
  lvars filed;

  sysopen("pthresh",0,"line") -> filed;
  datafile(filed) -> positive_threshold;
  sysclose(filed);
  sysopen("mathn",0,"line") -> filed;
  datafile(filed) -> N;
  sysclose(filed);
  sysopen("mathro",0,"line") -> filed;
  datafile(filed) -> ro;
  sysclose(filed);
  sysopen("mathk",0,"line") -> filed;
  datafile(filed) -> kappa;
  sysclose(filed);

enddefmethod;


;;; METHOD 13: check_ASSERT_UNASSERT_functions();
defmethod check_ASSERT_UNASSERT_functions;
  lvars ASSERT UNASSERT temp_list;
  lvars ASSERT_percent UNASSERT_percent;

  ;;; UNASSERT unsuccessful rules and reASSERT successful rules
  ;;;  that have previously been UNASSERTed

  (98 * N) / 100 -> ASSERT_percent;
  (2 * N) / 100 -> UNASSERT_percent;

  1 -> i;
  until i > number_of_rules
  do
    0 -> ASSERT;
    0 -> UNASSERT;

    rule_history(i) -> temp_list;
    until temp_list = []
    do
      if hd(temp_list) = 1
      then
        ASSERT + 1 -> ASSERT
      else
        UNASSERT + 1 -> UNASSERT
```

```
    endif;
    tl(temp_list) -> temp_list
   enduntil;

   if ASSERT > ASSERT_percent
   then
    "ASSERT" -> masking_of_rule(i)
   endif;

   if UNASSERT > UNASSERT_percent
   then
    "UNASSERT" -> masking_of_rule(i)
   endif;

   i+1 -> i
  enduntil;

  ;;;  METHOD 14)
  ^save_to_files_2;
  ;;; Reassign global variable to null value
  0 -> masking_of_rule;

enddefmethod;


;;; METHOD 14: save_to_files_2();
defmethod save_to_files_2;
 lvars filed;

 if type_of_structure = 'substrate_only'
 then
  syscreate("rmask1",1,"line") -> filed;
  masking_of_rule -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  syscreate("rmask2",1,"line") -> filed;
  masking_of_rule -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  syscreate("rmask3",1,"line") -> filed;
  masking_of_rule -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'MQW structure'
 then
  syscreate("rmask4",1,"line") -> filed;
  masking_of_rule -> datafile(filed);
  sysclose(filed);
 else
 endif;

enddefmethod;


;;; METHOD 15: fine_tuning_membership_functions
;;; FINE-TUNING AND CHANGING RULES TO DEAL WITH NEW FACTS
;;;
;;; SAMPLE OF 20 USED
defmethod fine_tuning_membership_functions;
 lvars i temp_list value variable_number temp2_list rule_number j;
 lvars tally_of_neg_dec;
 lvars range_of_negative_values member increment p q membership_function;
 lvars positive_highest positive_lowest median_value;
 lvars negative_highest temp3_list;
 lvars negative_lowest above below temp4_list variable_number2;
 lvars x sign_of_increment;
 lvars topone1 bottom2 top1 bot1 top2 bot2 y1 y2 height q width;

 ;;;  METHOD 16)
 ^retrieve_from_files_2;

 1 -> i;
 until i > 2
 do
  fuzzy_rules(i,1) -> temp_list;
 fuzzy_rules(i,2) -> temp4_list;
  until temp_list = []
  do
  if not(temp4_list = [])
  then
   if i = 2
   then
    hd(temp4_list) -> variable_number;
    tl(temp4_list) -> temp4_list;
    hd(temp4_list) -> variable_number2;
   else
    hd(temp4_list) -> variable_number;
   endif;
  endif;
 endif;
 ;;; CHECK IF RULE WAS USED IN THE CURRENT DECISION
 if ismember(hd(temp_list),list_of_rules_used)
 then
  if i = 2
  then
   ^choose_which_variable(variable_number, variable_number2) -> x;
   if x = 2
   then
    variable_number2 -> variable_number
   endif;
  endif;

  exp_variable(variable_number) -> value;
  ;;; CHECK IF OUTCOME OF DECISION WAS +VE OR -VE
  if outcome_of_decision = 1
  then
   ;;; ADD VALUE PASSED TO THE RULE TO A LIST OF +VE
   ;;; DECISIONS
   if length(variable_history(1,variable_number)) < 20
   then
    variable_history(1,variable_number) <> [^value]
     -> variable_history(1,variable_number)
   else
    tl(variable_history(1,variable_number)) <> [^value]
     -> variable_history(1,variable_number)
   endif;
  else
   ;;; ADD VALUE PASSED TO THE RULE TO A LIST
   ;;; OF -VE DECISIONS
   if length(variable_history(2,variable_number)) < 20
   then
    variable_history(2,variable_number) <> [^value]
     -> variable_history(2,variable_number)
   else
    tl(variable_history(2,variable_number)) <> [^value]
     -> variable_history(2,variable_number)
   endif;
  endif;

  ;;; PICK UP THE RULE HISTORY AND STORE IT IN temp2_list
  hd(temp_list) -> rule_number;
  rule_history(rule_number) -> temp2_list;
  0 -> tally_of_neg_dec;
  1 -> j;
  ;;; COUNT THE NUMBER OF -VE DECISIONS INVOLVING
  ;;; THIS RULE
  until temp2_list = []
  do
   if hd(temp2_list) = 0
   then
    tally_of_neg_dec+1 -> tally_of_neg_dec
   endif;
   tl(temp2_list) -> temp2_list;
  enduntil;

  ;;; THRESHOLD SET FOR NEGATIVE DECISIONS
  if tally_of_neg_dec > 18
  then
   if rule_variable(variable_number) = 'NONE'
   then
    1 -> member;
   elseif rule_variable(variable_number) = 'SOME'
   then
    2 -> member;
   elseif rule_variable(variable_number) = 'FAIRLY'
   then
    3 -> member;
   elseif rule_variable(variable_number) = 'VERY'
   then
    4 -> member;
   else
    5 -> member;
   endif;

   ;;; GET THE MEMBERSHIP FUNCTION FOR THIS VARIABLE
   mf(variable_number) -> q;
   membership_functions(q) -> membership_function;
```

```
variable_history(1,variable_number) -> temp3_list;
get_median_value(temp3_list) -> median_value;
0.0 -> positive_highest;
1.0 -> positive_lowest;
until temp3_list = []
do
  max(hd(temp3_list),positive_highest) -> positive_highest;
  min(hd(temp3_list),positive_lowest) -> positive_lowest;
  tl(temp3_list) -> temp3_list
enduntil;

variable_history(2,variable_number) -> temp3_list;
0.0 -> range_of_negative_values;
0.0 -> negative_highest;
1.0 -> negative_lowest;
0 -> above;
0 -> below;
until temp3_list = []
do
  max(hd(temp3_list),negative_highest) -> negative_highest;
  min(hd(temp3_list),negative_lowest) -> negative_lowest;
  if hd(temp3_list) > (positive_highest -
       ((positive_highest - median_value) * 20/100))
  then
    above + 1 -> above
  elseif hd(temp3_list) < (positive_lowest +
       ((median_value - positive_lowest) * 20/100))
  then
    below + 1 -> below
  else
  endif;
  tl(temp3_list) -> temp3_list
enduntil;
negative_highest - negative_lowest -> range_of_negative_values;

if above > 16 and member < 5
then
  ;;; DECREASE UPPER BOUNDARY OF
  ;;; MEMBERSHIP FUNCTION
  member * 2 -> p;
  membership_function(p) -> topone1;
  (0.5 - abs(0.5 - membership_function(p))) *
        (1 / (range_of_negative_values * 10)) -> increment;
  membership_function(p) - increment -> membership_function(p);

  membership_function(p) -> top1;
  (membership_function(p) + membership_function(p-1))/2 -> bot1;

  member + 1 -> member;
  ;;; DECREASE LOWER BOUND OF NEXT
  ;;; MEMBERSHIP FUNCTION BY THE SAME AMOUNT
  (member * 2) - 1 -> p;
  membership_function(p) -> bottom2;
  membership_function(p) - increment -> membership_function(p);

  (membership_function(p) + membership_function(p+1))/2 -> top2;
  membership_function(p) -> bot2;

  ((bot1 - ((top1 + bot2)/2)) / (top1 - bot1)) + 1 -> y1;
  ((((top1 + bot2)/2) - bot2) / (top2 - bot2) -> y2;
  y1 + y2 -> height;

  if height > 1.0
  then
    (member - 1) * 2 -> p;
    topone1 -> membership_function(p);
    (member * 2) - 1 -> p;
    bottom2 -> membership_function(p);
  endif;

  (member - 1) * 2 -> p;
  (member * 2) - 1 -> q;
  (membership_function(q+1) - membership_function(q)) -> width;

  if membership_function(p) > (membership_function(q) - (width/10))
  then
    (membership_function(q) - (width/10)) -> membership_function(p)
  endif;

  if membership_function(p) < (membership_function(q) - (width/2))
  then
    (membership_function(q) - (width/2)) -> membership_function(p)
  endif;

  membership_function -> membership_functions(variable_number);

  1 -> sign_of_increment;
endif;


if below > 16 and member > 1
then
  ;;; INCREASE LOWER BOUNDARY OF
  ;;; MEMBERSHIP FUNCTION
  (member * 2) - 1 -> p;
  membership_function(p) -> bottom2;
  (0.5 - abs(0.5 - membership_function(p))) *
        (1 / (range_of_negative_values * 10)) -> increment;
  membership_function(p) + increment -> membership_function(p);

  (membership_function(p) + membership_function(p+1))/2 -> top2;
  membership_function(p) -> bot2;

  member - 1 -> member;
  ;;; INCREASE UPPER BOUND OF THE PREVIOUS
  ;;; MEMBERSHIP FUNCTION
  ;;; BY THE SAME AMOUNT
  member * 2 -> p;
  membership_function(p) -> topone1;
  membership_function(p) + increment -> membership_function(p);

  membership_function(p) -> top1;
  (membership_function(p) + membership_function(p-1))/2 -> bot1;

  ((bot1 - ((top1 + bot2)/2)) / (top1 - bot1)) + 1 -> y1;
  ((((top1 + bot2)/2) - bot2) / (top2 - bot2) -> y2;
  y1 + y2 -> height;

  if height > 1.0
  then
    member * 2 -> p;
    topone1 -> membership_function(p);
    ((member + 1) * 2) - 1 -> p;
    bottom2 -> membership_function(p);
  endif;

  ((member + 1) * 2) - 1 -> p;
  member * 2 -> q;
  (membership_function(q) - membership_function(q-1)) -> width;

  if membership_function(p) > (membership_function(q) + (width/10))
  then
    (membership_function(q) + (width/10)) -> membership_function(p)
  endif;

  if membership_function(p) < (membership_function(q) + (width/2))
  then
    (membership_function(q) + (width/2)) -> membership_function(p)
  endif;

  membership_function -> membership_functions(variable_number);

  2 -> sign_of_increment
endif;

if sign_of_increment = 1
then
  var_changes(1,variable_number) <> [^increment]
      -> var_changes(1,variable_number);
  member - 1 -> member;
  var_changes(2,variable_number) <> [^member]
      -> var_changes(2,variable_number)
else
  (-1.0) * increment -> increment;
  var_changes(1,variable_number) <> [^increment]
      -> var_changes(1,variable_number);
  member + 1 -> member;
  var_changes(2,variable_number) <> [^member]
      -> var_changes(2,variable_number)
endif;
endif;
endif;

tl(temp4_list) -> temp4_list;
  tl(temp_list) -> temp_list;
enduntil;
i + 1 -> i
enduntil;
```

```
  ^save_to_files_1;
  ;;; METHOD 18)
  ^save_to_files_3;
enddefmethod;


;;; METHOD 16: retrieve_from_files_2();
defmethod retrieve_from_files_2;
 lvars filed;

 if type_of_structure = 'substrate only'
 then
  sysopen("frules1",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist1",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch1",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  sysopen("frules2",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist2",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch2",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
  sysopen("frules3",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist3",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch3",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
 else
  sysopen("frules4",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist4",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch4",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
 endif;

enddefmethod;


;;; METHOD 17: choose_which_variable(number1, number2);
defmethod choose_which_variable(number1, number2);
 lvars number1 number2 temp_list count number;
 lvars value rule_value membership member;
 lvars start finish halfway midpoint output_value;
 lvars first_value x;

 [] -> temp_list;
 temp_list <> [^number1] -> temp_list;
 temp_list <> [^number2] -> temp_list;
 1 -> count;

 until temp_list = []
 do
  hd(temp_list) -> number;

  exp_variable(number) -> value;
  rule_variable(number) -> rule_value;
  membership_functions(mf(number)) -> membership;

  if rule_value = 'NONE'
  then
   1 -> member
  elseif rule_value = 'SOME'
  then
   2 -> member
  elseif rule_value = 'FAIRLY'
  then
   3 -> member
  elseif rule_value = 'VERY'
  then
   4 -> member
  else
   5 -> member
  endif;

  (member * 2) - 1 -> member;
  membership(member) -> start;
  member + 1 -> member;
  membership(member) -> finish;

  (start - finish)/2.0 -> halfway;
  start + halfway -> midpoint;

  if value < midpoint
  then
   (value - start) / (midpoint - start) -> output_value
  else
   ((midpoint - value) / (finish - midpoint)) + 1 -> output_value
  endif;

  if output_value > 1.0 or output_value < 0.0
  then
   0.0 -> output_value
  endif;

  if count = 1
  then
   output_value -> first_value
  endif;

  tl(temp_list) -> temp_list;
  count + 1 -> count
 enduntil;

 if output_value < first_value
 then
  2 -> x
 else
  1 -> x
 endif;

 return(x)

enddefmethod;


;;; METHOD 18: save_to_files_3();
defmethod save_to_files_3;
 lvars filed;

 if type_of_structure = 'substrate only'
 then
  syscreate("rhist1",1,"line") -> filed;
  rule_history -> datafile(filed);
  sysclose(filed);
  syscreate("memfunc1",1,"line") -> filed;
  membership_functions -> datafile(filed);
  sysclose(filed);
  syscreate("memf1",1,"line") -> filed;
  mf -> datafile(filed);
  sysclose(filed);
  syscreate("varhist1",1,"line") -> filed;
  variable_history -> datafile(filed);
  sysclose(filed);
  syscreate("varch1",1,"line") -> filed;
  var_changes -> datafile(filed);
  sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
  syscreate("rhist2",1,"line") -> filed;
  rule_history -> datafile(filed);
  sysclose(filed);
  syscreate("memfunc2",1,"line") -> filed;
  membership_functions -> datafile(filed);
  sysclose(filed);
  syscreate("memf2",1,"line") -> filed;
  mf -> datafile(filed);
  sysclose(filed);
  syscreate("varhist2",1,"line") -> filed;
```

```
    variable_history -> datafile(filed);
    sysclose(filed);
    syscreate("varch2",1,"linc") -> filed;
    var_changes -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'multiple layers'
  then
    syscreate("rhist3",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
    syscreate("memfunc3",1,"line") -> filed;
    membership_functions -> datafile(filed);
    sysclose(filed);
    syscreate("memf3",1,"linc") -> filed;
    mf -> datafile(filed);
    sysclose(filed);
    syscreate("varhist3",1,"linc") -> filed;
    variable_history -> datafile(filed);
    sysclose(filed);
    syscreate("varch3",1,"line") -> filed;
    var_changes -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'MQW structure'
  then
    syscreate("rhist4",1,"line") -> filed;
    rule_history -> datafile(filed);
    sysclose(filed);
    syscreate("memfunc4",1,"line") -> filed;
    membership_functions -> datafile(filed);
    sysclose(filed);
    syscreate("memf4",1,"line") -> filed;
    mf -> datafile(filed);
    sysclose(filed);
    syscreate("varhist4",1,"line") -> filed;
    variable_history -> datafile(filed);
    sysclose(filed);
    syscreate("varch4",1,"line") -> filed;
    var_changes -> datafile(filed);
    sysclose(filed);
  else
  endif;

  ;;; Reassign global variables to null values
  0 -> rule_variable;
  0 -> fuzzy_rules;
  0 -> rule_history;
  0 -> membership_functions;
  0 -> mf;
  0 -> variable_history;
  0 -> var_changes;

enddefmethod;


;;; METHOD 19: were_changes_made_to_credibility_weights();
defmethod were_changes_made_to_credibility_weights;
 lvars filed x;

 0 -> x;

 if type_of_structure = 'substrate only'
 then
   sysopen("angst1",0,"line") -> filed;
   datafile(filed) -> angst;
   sysclose(filed);
   syscreate("angst1",1,"line") -> filed;
   x -> datafile(filed);
   sysclose(filed);
 elseif type_of_structure = 'a single layer'
 then
   sysopen("angst2",0,"line") -> filed;
   datafile(filed) -> angst;
   sysclose(filed);
   syscreate("angst2",1,"line") -> filed;
   x -> datafile(filed);
   sysclose(filed);
 elseif type_of_structure = 'multiple layers'
 then
   sysopen("angst3",0,"line") -> filed;
   datafile(filed) -> angst;
   sysclose(filed);
   syscreate("angst3",1,"line") -> filed;
   x -> datafile(filed);
   sysclose(filed);
 else
   sysopen("angst4",0,"line") -> filed;
   datafile(filed) -> angst;
   sysclose(filed);
   syscreate("angst4",1,"line") -> filed;
   x -> datafile(filed);
   sysclose(filed);
 endif;

enddefmethod;


;;; METHOD 20: record_variable_values();
;;;
defmethod record_variable_values;
 lvars filed instance experimental_curve;
 lvars single_example_fuzzy_premise single_example_consequent i;
 lvars number_of_consequents consequent_tally;
 lvars consequent example_set_fuzzy_premise;
 lvars example_set_boolean_premise example_set_consequent;
 lvars variable_list variable_list2 single_example_boolean_premise;
 lvars example_set_boolean;

 if type_of_structure = 'substrate only'
 then
  sysopen("exper",0,"line") -> filed;
  datafile(filed) -> experimental_curve;
  sysclose(filed);

  [] -> single_example_fuzzy_premise;
  [] -> single_example_boolean_premise;
  [] -> single_example_consequent;

  [15 17 18 20] -> variable_list;
  [22 23 24 25] -> variable_list2;

  for i in variable_list
  do
   experimental_curve(i) -> instance;
   single_example_fuzzy_premise <> [^instance]
        -> single_example_fuzzy_premise;
  endfor;
  for i in variable_list2
  do
   experimental_curve(i) -> instance;
   single_example_boolean_premise <> [^instance]
        -> single_example_boolean_premise;
  endfor;

  sysopen("conrec1",0,"line") -> filed;
  datafile(filed) -> consequent_tally;
  sysclose(filed);

  7 -> number_of_consequents;
  1 -> i;
  until i > number_of_consequents
  do
   fuzzy_variable(i,6) -> consequent;
   single_example_consequent <> [^consequent]
      -> single_example_consequent;
   if consequent > 0.1
   then
    consequent_tally(i) + 1 -> consequent_tally(i)
   endif;
   i + 1 -> i
  enduntil;

  sysopen("pexset1",0,"line") -> filed;
  datafile(filed) -> example_set_fuzzy_premise;
  sysclose(filed);
  sysopen("bexset1",0,"line") -> filed;
  datafile(filed) -> example_set_boolean_premise;
  sysclose(filed);
  sysopen("cexset1",0,"line") -> filed;
  datafile(filed) -> example_set_consequent;
  sysclose(filed);

  [^single_example_fuzzy_premise] <> example_set_fuzzy_premise
          -> example_set_fuzzy_premise;
  [^single_example_boolean_premise] <> example_set_boolean_premise
        -> example_set_boolean_premise;
  [^single_example_consequent] <> example_set_consequent
        -> example_set_consequent;
```

```
 syscreate("pexset1",1,"line") -> filed;
 example_set_fuzzy_premise -> datafile(filed);
 sysclose(filed);
 syscreate("bexset1",1,"line") -> filed;
 example_set_boolean_premise -> datafile(filed);
 sysclose(filed);
 syscreate("cexset1",1,"line") -> filed;
 example_set_consequent -> datafile(filed);
 sysclose(filed);
 syscreate("conrec1",1,"line") -> filed;
 consequent_tally -> datafile(filed);
 sysclose(filed);

elseif type_of_structure = 'a single layer'
then
 sysopen("exper",0,"line") -> filed;
 datafile(filed) -> experimental_curve;
 sysclose(filed);

 [] -> single_example_fuzzy_premise;
 [] -> single_example_boolean_premise;
 [] -> single_example_consequent;

 [18 20 24 25 27 29] -> variable_list;
 [31 32 33 34 35 36 37 26 38 39 40 41 42 43 44 45] -> variable_list2;

 for i in variable_list
 do
  experimental_curve(i) -> instance;
  single_example_fuzzy_premise < > [^instance]
      -> single_example_fuzzy_premise;
 endfor;
 for i in variable_list2
 do
  experimental_curve(i) -> instance;
  single_example_boolean_premise < > [^instance]
      -> single_example_boolean_premise;
 endfor;

 sysopen("conrec2",0,"line") -> filed;
 datafile(filed) -> consequent_tally;
 sysclose(filed);

 18 -> number_of_consequents;
 1 -> i;
 until i > number_of_consequents
 do
  fuzzy_variable(i,6) -> consequent;
  single_example_consequent < > [^consequent]
      -> single_example_consequent;
  if consequent > 0.1
  then
   consequent_tally(i) + 1 -> consequent_tally(i)
  endif;
  i + 1 -> i
 enduntil;

 sysopen("pexset2",0,"line") -> filed;
 datafile(filed) -> example_set_fuzzy_premise;
 sysclose(filed);
 sysopen("bexset2",0,"line") -> filed;
 datafile(filed) -> example_set_boolean_premise;
 sysclose(filed);
 sysopen("cexset2",0,"line") -> filed;
 datafile(filed) -> example_set_consequent;
 sysclose(filed);

 [^single_example_fuzzy_premise] < > example_set_fuzzy_premise
      -> example_set_fuzzy_premise;
 [^single_example_boolean_premise] < > example_set_boolean_premise
      -> example_set_boolean_premise;
 [^single_example_consequent] < > example_set_consequent
      -> example_set_consequent;

 syscreate("pexset2",1,"line") -> filed;
 example_set_fuzzy_premise -> datafile(filed);
 sysclose(filed);
 syscreate("bexset2",1,"line") -> filed;
 example_set_boolean_premise -> datafile(filed);
 sysclose(filed);
 syscreate("cexset2",1,"line") -> filed;
 example_set_consequent -> datafile(filed);
 sysclose(filed);
 syscreate("conrec2",1,"line") -> filed;
 consequent_tally -> datafile(filed);
 sysclose(filed);

elseif type_of_structure = 'multiple layers'
then
 sysopen("exper",0,"line") -> filed;
 datafile(filed) -> experimental_curve;
 sysclose(filed);

 [] -> single_example_fuzzy_premise;
 [] -> single_example_boolean_premise;
 [] -> single_example_consequent;

 [15 17 20 22 27 29] -> variable_list;
 [30 31 32] -> variable_list2;

 for i in variable_list
 do
  single_example_fuzzy_premise < > [^experimental_curve(i)]
      -> single_example_fuzzy_premise;
 endfor;
 for i in variable_list2
 do
  single_example_boolean_premise < > [experimental_curve(i)]
      -> single_example_boolean_premise;
 endfor;

 sysopen("conrec3",0,"line") -> filed;
 datafile(filed) -> consequent_tally;
 sysclose(filed);

 7 -> number_of_consequents;
 1 -> i;
 until i > number_of_consequents
 do
  fuzzy_variable(i,6) -> consequent;
  single_example_consequent < > [^consequent]
      -> single_example_consequent;
  if consequent > 0.1
  then
   consequent_tally(i) + 1 -> consequent_tally(i)
  endif;
  i + 1 -> i
 enduntil;

 sysopen("pexset3",0,"line") -> filed;
 datafile(filed) -> example_set_fuzzy_premise;
 sysclose(filed);
 sysopen("bexset3",0,"line") -> filed;
 datafile(filed) -> example_set_boolean_premise;
 sysclose(filed);
 sysopen("cexset3",0,"line") -> filed;
 datafile(filed) -> example_set_consequent;
 sysclose(filed);

 [^single_example_fuzzy_premise] < > example_set_fuzzy_premise
      -> example_set_fuzzy_premise;
 [^single_example_boolean_premise] < > example_set_boolean_premise
      -> example_set_boolean_premise;
 [^single_example_consequent] < > example_set_consequent
      -> example_set_consequent;

 syscreate("pexset3",1,"line") -> filed;
 example_set_fuzzy_premise -> datafile(filed);
 sysclose(filed);
 syscreate("bexset3",1,"line") -> filed;
 example_set_boolean_premise -> datafile(filed);
 sysclose(filed);
 syscreate("cexset3",1,"line") -> filed;
 example_set_consequent -> datafile(filed);
 sysclose(filed);
 syscreate("conrec3",1,"line") -> filed;
 consequent_tally -> datafile(filed);
 sysclose(filed);

elseif type_of_structure = 'MQW structure'
then
 sysopen("exper",0,"line") -> filed;
 datafile(filed) -> experimental_curve;
 sysclose(filed);

 [] -> single_example_fuzzy_premise;
 [] -> single_example_boolean_premise;
 [] -> single_example_consequent;
```

```
  [14 16 15 25 27 30 31 33 57 59] -> variable_list;
  [41 42 21 23 43 44 45 60 46] -> variable_list2;

  for i in variable_list
  do
    single_example_fuzzy_premise < > [^experimental_curve(i)]
          -> single_example_fuzzy_premise;
  endfor;
  for i in variable_list2
  do
    single_example_boolean_premise < > [experimental_curve(i)]
          -> single_example_boolean_premise;
  endfor;

  sysopen("conrec4",0,"line") -> filed;
  datafile(filed) -> consequent_tally;
  sysclose(filed);

  14 -> number_of_consequents;
  1 -> i;
  until i > number_of_consequents
  do
    fuzzy_variable(i,6) -> consequent;
    single_example_consequent < > [^consequent]
          -> single_example_consequent;
    if consequent > 0.1
    then
      consequent_tally(i) + 1 -> consequent_tally(i)
    endif;
    i + 1 -> i
  enduntil;

  sysopen("pexset4",0,"line") -> filed;
  datafile(filed) -> example_set_fuzzy_premise;
  sysclose(filed);
  sysopen("bexset4",0,"line") -> filed;
  datafile(filed) -> example_set_boolean_premise;
  sysclose(filed);
  sysopen("cexset4",0,"line") -> filed;
  datafile(filed) -> example_set_consequent;
  sysclose(filed);

  [^single_example_fuzzy_premise] < > example_set_fuzzy_premise
          -> example_set_fuzzy_premise;
  [^single_example_boolean_premise] < > example_set_boolean_premise
          -> example_set_boolean_premise;
  [^single_example_consequent] < > example_set_consequent
          -> example_set_consequent;

  syscreate("pexset4",1,"line") -> filed;
  example_set_fuzzy_premise -> datafile(filed);
  sysclose(filed);
  syscreate("bexset4",1,"line") -> filed;
  example_set_boolean_premise -> datafile(filed);
  sysclose(filed);
  syscreate("cexset4",1,"line") -> filed;
  example_set_consequent -> datafile(filed);
  sysclose(filed);
  syscreate("conrec4",1,"line") -> filed;
  consequent_tally -> datafile(filed);
  sysclose(filed);
 else
 endif;

enddefmethod;


;;; ____________________________________

;;; METHOD 21: run_the_system();
;;; This method coordinates the control structures of the expert system
defmethod run_the_system();
  lvars continue frame_system experts combined_connection_matrix;
  lvars rules structural_parameters_frame value check value;

 nl(12);
 pr('_____________________________________________________');nl(2)
 pr('  FUZZY SYSTEM FOR X-RAY ROCKING
              CURVE ANALYSIS');nl(2);
 pr('_____________________________________________________');nl(3);
 pr('BRIEF DESCRIPTION OF THE SYSTEM:');nl(2);
 pr(' (1) The user describes an experimental rocking curve.');nl(3);
 pr(' (2) Fuzzy rules deduce a set of structural parameters from the');nl(1);
 pr('     the experimental curve.');nl(3);
 pr(' (3) These parameters are used to simulate a theoretical rocking');nl(1);
 pr('     curve using the RADS program. Input screens for the RADS');nl(1);
 pr('     program are shown.');nl(3);
 pr(' (4) The user describes the resulting theoretical rocking curve.');nl(3);
 pr(' (5) The experimental and theoretical rocking curves are compared');nl(1);
 pr('     and an objective measure of the closeness of fit is calculated.');nl(1);
 pr('     This measure is used to update the adaptive
                mechanisms of the system.');nl(3);

 pr('Press ENTER to continue ...');readline() -> value;nl(2);

 pr('_____________________________________________________');nl(2);

 pr('There are four main adaptive mechanisms: ');nl(2);
 pr('     (1) credibility weights and connection matrices
              switch the focus');nl(1);
 pr('         of attention of the system between different sets
              of fuzzy rules.');nl(2);
 pr('     (2) triangular membership functions of fuzzy variables
              are fine-tuned');nl(1);
 pr('         and changed in order to alter the meaning of fuzzy rules.');nl(2);
 pr('     (3) values from good decisions are recorded and used to create');nl(1);
 pr('         new fuzzy rules by inductive learning from examples.
              Each new');nl(1);
 pr('         fuzzy rule is rigorously tested using 6 fitness functions');nl(1);
 pr('         based on 6 evaluation criteria.');nl(2);
 pr('     (4) Rules that produce consistently incorrect decisions
              are UNASSERTed');nl(1);
 pr('         or removed from the reasoning processes of the system.');nl(1);
 pr('         These rules are ASSERTed again when they produce');nl(1);
 pr('         correct decisions.');nl(2);

 pr('Press ENTER to continue ...');readline() -> value;

 pr('_____________________________________________________');nl(3);

  ;;; METHOD 12)
  ^set_up_some_variables;

  [Y] -> continue;
  while continue=[Y] or continue=[y]
  do

 ;;; 1) MAIN BODY

 ;;; Create an instance of a frame system
 ;;; Setting up the frame system used to describe
 ;;; the experimental curve.
 make_instance([FRAME_SYSTEM]) -> frame_system;

 nl(6);
 pr('PLEASE DESCRIBE THE EXPERIMENTAL ROCKING CURVE ...');

 ;;; Ask the leading question and find out
 ;;; which type of structure is being analysed
 frame_system <- ask_the_leading_question;

 ;;; Initialize logic-based variables associated with this type of structure
 ;;; METHOD 1)
 ^initialise_rocking_curve_parameters(1);

 ;;; Fill that frame system using user responses to questions; and
 ;;; Transform all variables into a logic-based format.
 ;;; Then cancel the frame system.
 frame_system <- fill_the_frame_system(1);
 cancel frame_system;

 ;;; METHOD 2)
 ^store_rocking_curve_parameters(1);

 ;;; STORE DEFAULT VALUES IN THE STRUCTURAL PARAMETERS
 ;;;
 nl(2);pr('_____________________________________________________');nl(2);
 pr('THE CONTROL MECHANISMS CHOOSE A SET OF
      FUZZY RULES AND FIRE THESE RULES');nl(1);
 pr(' TO INFER THE STRUCTURAL PARAMETERS');nl(3);
 pr('The setfuzz.p program stores default values into
      structural parameters.');nl(2);
 load setfuzz.p;

 ;;; Create a Combined Connection Matrix
 ;;;    called combined_connection_matrix
 make_instance([COMBINED_MATRIX]) -> combined_connection_matrix;
```

```
;;; The values of this new matrix will be taken
;;;   from connection matrices for each of the experts

;;; METHOD 19)
^were_changes_made_to_credibility_weights;

if angst = 1
then
 ;;; METHOD 3)
 ^create_the_connection_matrices -> experts;

 combined_connection_matrix <- create(experts);
 cancel experts;
else
 combined_connection_matrix <- retrieve_from_file;
endif;

0 -> angst;

;;; METHOD 4)
^load_the_rules_from_file;
;;; Rules are stored in the global variable 'rule'

;;; Make deductions based on those rules with the highest weightings
;;; in the Combined Connection Matrix
;;; The list of rules used is passed back ... to list_of_rules_used

nl(2);pr('Firing those rules with the highest weights
        in the combined connection matrix');nl(2);
combined_connection_matrix <- make_deductions -> list_of_rules_used;
combined_connection_matrix <- save_to_file;
cancel combined_connection_matrix;

nl(2);
make_instance([STRUCTURAL_PARAMETERS_FRAME])
      -> structural_parameters_frame;
structural_parameters_frame<-fill_the_frame;
;;; DISPLAY CONCLUSIONS AS SCREEN OUTPUT
pr('_____ _____');nl(2);

structural_parameters_frame=>
cancel structural_parameters_frame;

;;; Create an instance of a frame system
;;; Setting up the frame system used to describe
;;;  the simulated curve.
make_instance([FRAME_SYSTEM]) -> frame_system;

;;; Initialize logic-based variables associated with this type of structure
;;; Initializing the parameters used to describe the simulated rocking curve

;;; METHOD 1)
^initialise_rocking_curve_parameters(2);

nl(3);
pr('PLEASE DESCRIBE THE SIMULATED ROCKING CURVE...');nl(3);

;;; Fill that frame system using user responses to questions; and
;;; Transform all variables into a logic-based format.
;;; Then cancel the frame system.
frame_system <- fill_the_frame_system(2);
cancel frame_system;

pr('_____');nl(3);
;;; METHOD 2)
^store_rocking_curve_parameters(2);


;;; OUTCOME OF DECISION

pr('Calculating the outcome of the decision');nl(2);
;;;  METHOD 5)
^calculate_the_outcome -> outcome_of_decision;

nl(4);
pr('CALCULATED OUTCOME OF DECISION : ');
pr(outcome_of_decision);nl(2);
pr('Value is between 0 and 1 : threshold of 0.8 is
        currently set for SUCCESS');nl(2);

if outcome_of_decision > positive_threshold
then
 ;;; METHOD 20)
 ^record_variable_values;
endif;

;;; Outcome of Decision is recorded as
;;; 1 = Successful or 0 = Unsuccessful
if (outcome_of_decision > 0.8)
then
   1 -> outcome_of_decision
elseif (outcome_of_decision < 0.78)
then
   0 -> outcome_of_decision
else
 ;;; METHOD 6)
 ^use_the_unasserted_rules;
 0 -> outcome_of_decision
endif;

pr('The outcome of this expert system decision was ');
if outcome_of_decision = 1
then
 pr('SUCCESS')
else
 pr('FAILURE')
endif;pr('.');nl(2);
pr('The following rules were used in this decision :');nl(2);
ppr(list_of_rules_used);nl(2);
pr('Press ENTER to continue ...');readline() -> value;

;;; METHOD 7)
^update_the_histories -> experts;

;;; METHOD 10)
^check_experts_credibility_weights(experts);
cancel experts;

;;; METHOD 13)
^check_ASSERT_UNASSERT_functions;

;;; METHOD 15)
^fine_tuning_membership_functions;

pr('_____');

0 -> check;
while (check=0)
do
 nl(3);pr('Do you wish to describe another rocking curve (Y/N) ');
 readline() -> continue;
 test_string(continue) -> check
endwhile;


if continue=[Y] or continue=[y]
then
 nl(2);pr('Please WAIT while the system is reinitialised ...');nl(7);
endif;

 endwhile;

enddefmethod;


endflavour;
```

[3] The program 'setfuzz.p' sets up the fuzzy consequent variables.

```
;;; THIS PROGRAM SETS UP THE FUZZY OUTPUT VARIABLES
;;; IT IS CALLED FROM:
;;;   OBJECT: [FUZZY_SYSTEM] METHOD: [run_the_system]
;;;
;;; First an array is set up to store the output values
;;;  This is an mx7 array, where m is the number of fuzzy output variables
;;;  The first five columns store the value calculated from the premise for
;;;    each of 5 output membership functions
;;;  Column 7 stores the membership functions (in list form) of the output
;;;    variable
;;;  Column 6 stores the discrete output value that will be calculated using the
;;;    Centroid Defuzzification Method

if type_of_structure = 'substrate only'
then

 newarray([1 7 1 7]) -> fuzzy_variable;

 for i in [1 2 3 4 5 6 7]
 do
  for j in [1 2 3 4 5]
  do
   0.0 -> fuzzy_variable(i,j);
  endfor;
 endfor;

 ;;; Number 1
 0 -> crystal_quality;
 crystal_quality -> fuzzy_variable(1,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1,7);

 ;;; Number 2
 0 -> strain_in_surface_layer_of_sample;
 strain_in_surface_layer_of_sample -> fuzzy_variable(2,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2,7);

 ;;; Number 3
 0 -> reference_crystal_is_different_to_substrate;
 reference_crystal_is_different_to_substrate -> fuzzy_variable(3,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3,7);

 ;;; Number 4
 0 -> crystal_consists_of_subgrains;
 crystal_consists_of_subgrains -> fuzzy_variable(4,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4,7);

 ;;; Number 5
 0 -> characteristic_curve;
 characteristic_curve -> fuzzy_variable(5,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5,7);

 ;;; Number 6
 0 -> misorientation_of_substrate;
 misorientation_of_substrate -> fuzzy_variable(6,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6,7);

 ;;; Number 7
 0 -> incorrect_experiment;
 incorrect_experiment -> fuzzy_variable(7,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7,7);

elseif type_of_structure = 'a single layer'
then

 newarray([1 18 1 7]) -> fuzzy_variable;

 for i in [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18]
 do
  for j in [1 2 3 4 5]
  do
   0.0 -> fuzzy_variable(i,j);
  endfor;
 endfor;

 ;;; Number 1
 0 -> crystal_quality;
 crystal_quality -> fuzzy_variable(1,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1,7);

 ;;; Number 2
 0 -> characteristic_curve;
 characteristic_curve -> fuzzy_variable(2,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2,7);

 ;;; Number 3
 0 -> misorientation_of_substrate;
 misorientation_of_substrate -> fuzzy_variable(3,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3,7);

 ;;; Number 4
 0 -> bending_of_substrate;
 bending_of_substrate -> fuzzy_variable(4,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4,7);

 ;;; Number 5
 0 -> grading_of_the_layer;
 grading_of_the_layer -> fuzzy_variable(5,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5,7);

 ;;; Number 6
 0 -> layer_is_thick;
 layer_is_thick -> fuzzy_variable(6,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6,7);

 ;;; Number 7
 0 -> change_in_lattice_parameter_with_depth;
 change_in_lattice_parameter_with_depth -> fuzzy_variable(7,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7,7);

 ;;; Number 8
 0 -> layer_is_present_in_the_substrate_peak;
 layer_is_present_in_the_substrate_peak -> fuzzy_variable(8,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8,7);

 ;;; Number 9
 0 -> layer_is_thin;
 layer_is_thin -> fuzzy_variable(9,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9,7);

 ;;; Number 10
 0 -> evidence_of_mismatch;
 evidence_of_mismatch -> fuzzy_variable(10,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10,7);

 ;;; Number 11
 0 -> peak_is_outside_the_scan_range;
 peak_is_outside_the_scan_range -> fuzzy_variable(11,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(11,7);

 ;;; Number 12
 0 -> misoriented_or_mismatched_layer;
 misoriented_or_mismatched_layer -> fuzzy_variable(12,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(12,7);

 ;;; Number 13
 0 -> multiple_layers;
 multiple_layers -> fuzzy_variable(13,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(13,7);

 ;;; Number 14
 0 -> simulation_or_calibration_chart_is_needed;
 simulation_or_calibration_chart_is_needed -> fuzzy_variable(14,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(14,7);

 ;;; Number 15
 0 -> relaxation;
 relaxation -> fuzzy_variable(15,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(15,7);

 ;;; Number 16
 0 -> incorrect_experiment;
 incorrect_experiment -> fuzzy_variable(16,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(16,7);

 ;;; Number 17
 0 -> thickness_of_layer;
 incorrect_experiment -> fuzzy_variable(17,6);
 [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(17,7);
```

```
    ;;; Number 18
    0 -> experimental_mismatch;
    incorrect_experiment -> fuzzy_variable(18,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(18,7);

  elseif type_of_structure = 'multiple layers'
  then

    newarray([1 10 1 7]) -> fuzzy_variable;

    for i in [1 2 3 4 5 6 7 8 9 10]
    do
      for j in [1 2 3 4 5]
      do
        0.0 -> fuzzy_variable(i,j);
      endfor;
    endfor;

    ;;; Number 1
    0 -> crystal_quality;
    crystal_quality -> fuzzy_variable(1,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1,7);

    ;;; Number 2
    0 -> misorientation_of_substrate;
    misorientation_of_substrate -> fuzzy_variable(2,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2,7);

    ;;; Number 3
    0 -> simulation_or_calibration_chart_is_needed;
    simulation_or_calibration_chart_is_needed -> fuzzy_variable(3,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3,7);

    ;;; Number 4
    0 -> relaxation;
    relaxation -> fuzzy_variable(4,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4,7);

    ;;; Number 5
    0 -> layers_are_thick;
    layers_are_thick -> fuzzy_variable(5,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5,7);

    ;;; Number 6
    0 -> there_are_hidden_layers_somewhere;
    there_are_hidden_layers_somewhere -> fuzzy_variable(6,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6,7);

    ;;; Number 7
    0 -> incorrect_experiment;
    incorrect_experiment -> fuzzy_variable(7,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7,7);

    ;;; Number 8
    0 -> evidence_of_mismatch;
    evidence_of_mismatch -> fuzzy_variable(8,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8,7);

    ;;; Number 9
    0 -> there_are_thin_layers_at_the_interfaces;
    there_are_thin_layers_at_the_interfaces -> fuzzy_variable(9,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9,7);

    ;;; Number 10
    0 -> evidence_of_interferometer_structure;
    evidence_of_interferometer_structure -> fuzzy_variable(10,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10,7);

    0.0 -> grading_of_the_layer;
    0.0 -> layer_is_thick;

  else
    newarray([1 14 1 7]) -> fuzzy_variable;

    for i in [1 2 3 4 5 6 7 8 9 10 11 12 13 14]
    do
      for j in [1 2 3 4 5]
      do
        0.0 -> fuzzy_variable(i,j);
      endfor;
    endfor;

    ;;; Number 1
    0 -> crystal_quality;
    crystal_quality -> fuzzy_variable(1,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1,7);

    ;;; Number 2
    0 -> characteristic_curve;
    characteristic_curve -> fuzzy_variable(2,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2,7);

    ;;; Number 3
    0 -> misorientation_of_substrate;
    misorientation_of_substrate -> fuzzy_variable(3,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3,7);

    ;;; Number 4
    0 -> simulation_or_calibration_chart_is_needed;
    simulation_or_calibration_chart_is_needed -> fuzzy_variable(4,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4,7);

    ;;; Number 5
    0 -> relaxation;
    relaxation -> fuzzy_variable(5,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5,7);

    ;;; Number 6
    0 -> layers_are_thick;
    layers_are_thick -> fuzzy_variable(6,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6,7);

    ;;; Number 7
    0 -> layers_are_thin;
    layers_are_thin -> fuzzy_variable(7,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7,7);

    ;;; Number 8
    0 -> grading_or_dispersion_of_layer_thicknesses;
    grading_or_dispersion_of_layer_thicknesses -> fuzzy_variable(8,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8,7);

    ;;; Number 9
    0 -> grading_occurs_through_AB_layers;
    grading_occurs_through_AB_layers -> fuzzy_variable(9,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9,7);

    ;;; Number 10
    0 -> large_overall_layer_thickness;
    large_overall_layer_thickness -> fuzzy_variable(10,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10,7);

    ;;; Number 11
    0 -> layers_are_not_uniform;
    layers_are_not_uniform -> fuzzy_variable(11,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(11,7);

    ;;; Number 12
    0 -> incorrect_experiment;
    incorrect_experiment -> fuzzy_variable(12,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(12,7);

    ;;; Number 13
    0 -> period_of_superlattice;
    incorrect_experiment -> fuzzy_variable(13,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(13,7);

    ;;; Number 14
    0 -> period_dispersion;
    incorrect_experiment -> fuzzy_variable(14,6);
    [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(14,7);

    0.0 -> evidence_of_mismatch;
    0.0 -> grading_of_the_layer;
    0.0 -> layer_is_thick;

  endif;

  ;;; END OF PROGRAM
  ;;; ____________________________________
```

[4] The program 'rules1.p' uses the numerical representation to set up the rules for the first partition of the fuzzy rulebase.

```
;;; PROGRAM rules1.p

;;; This program sets up the rules for substrate only structure
;;; PROGRAM IS CALLED FROM
;;;   OBJECT [FUZZY_SYSTEM] METHOD [load_the_rules_from_file]

sysopen("rno1",0,"line") -> filed;
datafile(filed) -> number_of_rules;
sysclose(filed);

sysopen("rules1",0,"line") -> filed;
datafile(filed) -> existing_rules;
sysclose(filed);

sysopen("exper",0,"line") -> filed;
datafile(filed) -> dummy_array;
sysclose(filed);

sysopen("frules1",0,"line") -> filed;
datafile(filed) -> fuzzy_rules;
sysclose(filed);

sysopen("memfunc1",0,"line") -> filed;
datafile(filed) -> membership_functions;
sysclose(filed);

sysopen("memf1",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);

sysopen("fc1",0,"line") -> filed;
datafile(filed) -> consequences;
sysclose(filed);

sysopen("l1",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);


newarray([1 4]) -> fp;
newarray([1 4]) -> bp;

;;; FUZZY PREMISES
;;; substrate_peak_broadening
dummy_array(15) -> fp(1);
;;; substrate_asymmetry_in_peak
dummy_array(17) -> fp(2);
;;; interference_fringes
dummy_array(18) -> fp(3);
;;; visibility_of_interference_fringes
dummy_array(20) -> fp(4);
;;; BOOLEAN PREMISES
;;; type_of_structure_is_substrate_only
dummy_array(22) -> bp(1);
;;; number_of_peaks_is_one
dummy_array(23) -> bp(2);
;;; number_of_peaks_is_more_than_one
dummy_array(24) -> bp(3);
;;; number_of_peaks_is_none
dummy_array(25) -> bp(4);

load rules.p

 newarray([1 ^number_of_fuzzy_variables]) -> exp_variable;

 substrate_peak_broadening -> exp_variable(1);
 substrate_peak_broadening -> exp_variable(2);
 substrate_peak_broadening -> exp_variable(3);
 substrate_peak_broadening -> exp_variable(4);
 substrate_peak_broadening -> exp_variable(5);
 substrate_asymmetry_in_peak -> exp_variable(6);
 substrate_asymmetry_in_peak -> exp_variable(7);
 substrate_asymmetry_in_peak -> exp_variable(8);
 substrate_asymmetry_in_peak -> exp_variable(9);
 substrate_asymmetry_in_peak -> exp_variable(10);
 substrate_peak_broadening -> exp_variable(11);
 substrate_peak_broadening -> exp_variable(12);
 substrate_peak_broadening -> exp_variable(13);
 substrate_peak_broadening -> exp_variable(14);
 substrate_peak_broadening -> exp_variable(15);

;;; END of program.
```

[5] The program 'rules2.p' uses the numerical representation to set up the rules for the second partition of the fuzzy rulebase.

```
;;; PROGRAM rules2.p

;;; This program sets up the rules for single layer structure
;;; PROGRAM IS CALLED FROM
;;;   OBJECT [FUZZY_SYSTEM] METHOD [load_the_rules_from_file]

sysopen("rno2",0,"line") -> filed;
datafile(filed) -> number_of_rules;
sysclose(filed);

sysopen("rules2",0,"line") -> filed;
datafile(filed) -> existing_rules;
sysclose(filed);

sysopen("exper",0,"line") -> filed;
datafile(filed) -> dummy_array;
sysclose(filed);

sysopen("frules2",0,"line") -> filed;
datafile(filed) -> fuzzy_rules;
sysclose(filed);

sysopen("memfunc2",0,"line") -> filed;
datafile(filed) -> membership_functions;
sysclose(filed);

sysopen("memf2",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);

sysopen("fc2",0,"line") -> filed;
datafile(filed) -> consequences;
sysclose(filed);

sysopen("l2",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);


newarray([1 8]) -> fp;
newarray([1 16]) -> bp;

;;; FUZZY PREMISES
;;; substrate_peak_broadening
dummy_array(18) -> fp(1);
;;; substrate_asymmetry_in_peak
dummy_array(20) -> fp(2);
;;; layer_asymmetry_in_peak
dummy_array(24) -> fp(3);
;;; layer_wedge_shaped_peak
dummy_array(25) -> fp(4);
;;; interference_fringes
dummy_array(27) -> fp(5);
;;; visibility_of_interference_fringes
dummy_array(29) -> fp(6);
;;; intensity_of_layer_peak
dummy_array(47) -> fp(7);
;;; evidence_of_mismatch
evidence_of_mismatch -> fp(8);

;;; BOOLEAN PREMISES
;;; type_of_structure_is_single_layer
dummy_array(31) -> bp(1);
;;; number_of_peaks_is_one
dummy_array(32) -> bp(2);
;;; number_of_peaks_is_two
dummy_array(33) -> bp(3);
;;; number_of_peaks_is_more_than_two
dummy_array(34) -> bp(4);
;;; number_of_peaks_is_none
dummy_array(35) -> bp(5);
;;; substrate_material_equal_to_layer
dummy_array(36) -> bp(6);
;;; peak_splitting_less_than_150
dummy_array(37) -> bp(7);
;;; layer_split_peak
dummy_array(26) -> bp(8);
;;; layer_integrated_intensity_of_peak_is_zero
dummy_array(38) -> bp(9);
;;; layer_thickness_greater_than_half_micron
dummy_array(39) -> bp(10);
;;; layer_thickness_less_than_5_microns
dummy_array(40) -> bp(11);
;;; peak_splitting_is_zero
dummy_array(41) -> bp(12);
;;; peak_splitting_less_than_three_times_width_of_peak
dummy_array(42) -> bp(13);
;;; relaxed_mismatch_is_high
dummy_array(43) -> bp(14);
;;; peak_splitting_is_high
dummy_array(44) -> bp(15);
;;; spacing_of_interference_fringes_is_low
dummy_array(45) -> bp(16);

load rules.p

 newarray([1 ^number_of_fuzzy_variables])->exp_variable;
 substrate_peak_broadening -> exp_variable(1);
 substrate_peak_broadening -> exp_variable(2);
 substrate_peak_broadening -> exp_variable(3);
 substrate_peak_broadening -> exp_variable(4);
 substrate_peak_broadening -> exp_variable(5);
 layer_asymmetry_in_peak -> exp_variable(6);
 layer_wedge_shaped_peak -> exp_variable(7);
 layer_asymmetry_in_peak -> exp_variable(8);
 layer_wedge_shaped_peak -> exp_variable(9);
 layer_asymmetry_in_peak -> exp_variable(10);
 layer_wedge_shaped_peak -> exp_variable(11);
 layer_asymmetry_in_peak -> exp_variable(12);
 layer_wedge_shaped_peak -> exp_variable(13);
 layer_asymmetry_in_peak -> exp_variable(14);
 layer_wedge_shaped_peak -> exp_variable(15);
 layer_asymmetry_in_peak -> exp_variable(16);
 layer_wedge_shaped_peak -> exp_variable(17);
 layer_asymmetry_in_peak -> exp_variable(18);
 layer_wedge_shaped_peak -> exp_variable(19);
 layer_asymmetry_in_peak -> exp_variable(20);
 layer_wedge_shaped_peak -> exp_variable(21);
 layer_asymmetry_in_peak -> exp_variable(22);
 layer_wedge_shaped_peak -> exp_variable(23);
 layer_asymmetry_in_peak -> exp_variable(24);
 layer_wedge_shaped_peak -> exp_variable(25);
 layer_asymmetry_in_peak -> exp_variable(26);
 layer_wedge_shaped_peak -> exp_variable(27);
 layer_asymmetry_in_peak -> exp_variable(28);
 layer_wedge_shaped_peak -> exp_variable(29);
 layer_asymmetry_in_peak -> exp_variable(30);
 layer_wedge_shaped_peak -> exp_variable(31);
 layer_asymmetry_in_peak -> exp_variable(32);
 layer_wedge_shaped_peak -> exp_variable(33);
 layer_asymmetry_in_peak -> exp_variable(34);
 layer_wedge_shaped_peak -> exp_variable(35);
 layer_asymmetry_in_peak -> exp_variable(36);
 layer_wedge_shaped_peak -> exp_variable(37);
 layer_asymmetry_in_peak -> exp_variable(38);
 layer_wedge_shaped_peak -> exp_variable(39);
 layer_asymmetry_in_peak -> exp_variable(40);
 layer_wedge_shaped_peak -> exp_variable(41);
 layer_asymmetry_in_peak -> exp_variable(42);
 layer_wedge_shaped_peak -> exp_variable(43);
 layer_asymmetry_in_peak -> exp_variable(44);
 layer_wedge_shaped_peak -> exp_variable(45);
 layer_asymmetry_in_peak -> exp_variable(46);
 layer_wedge_shaped_peak -> exp_variable(47);
 layer_asymmetry_in_peak -> exp_variable(48);
 layer_wedge_shaped_peak -> exp_variable(49);
 layer_asymmetry_in_peak -> exp_variable(50);
 layer_wedge_shaped_peak -> exp_variable(51);
 layer_asymmetry_in_peak -> exp_variable(52);
 layer_wedge_shaped_peak -> exp_variable(53);
 layer_asymmetry_in_peak -> exp_variable(54);
 layer_wedge_shaped_peak -> exp_variable(55);
 interference_fringes -> exp_variable(56);
 visibility_of_interference_fringes -> exp_variable(57);
```

```
interference_fringes -> exp_variable(58);
visibility_of_interference_fringes -> exp_variable(59);
interference_fringes -> exp_variable(60);
visibility_of_interference_fringes -> exp_variable(61);
interference_fringes -> exp_variable(62);
visibility_of_interference_fringes -> exp_variable(63);
interference_fringes -> exp_variable(64);
visibility_of_interference_fringes -> exp_variable(65);
interference_fringes -> exp_variable(66);
visibility_of_interference_fringes -> exp_variable(67);
interference_fringes -> exp_variable(68);
visibility_of_interference_fringes -> exp_variable(69);
interference_fringes -> exp_variable(70);
visibility_of_interference_fringes -> exp_variable(71);
interference_fringes -> exp_variable(72);
visibility_of_interference_fringes -> exp_variable(73);
interference_fringes -> exp_variable(74);
visibility_of_interference_fringes -> exp_variable(75);
interference_fringes -> exp_variable(76);
visibility_of_interference_fringes -> exp_variable(77);
interference_fringes -> exp_variable(78);
visibility_of_interference_fringes -> exp_variable(79);
interference_fringes -> exp_variable(80);
visibility_of_interference_fringes -> exp_variable(81);
interference_fringes -> exp_variable(82);
visibility_of_interference_fringes -> exp_variable(83);
interference_fringes -> exp_variable(84);
visibility_of_interference_fringes -> exp_variable(85);
interference_fringes -> exp_variable(86);
visibility_of_interference_fringes -> exp_variable(87);
interference_fringes -> exp_variable(88);
visibility_of_interference_fringes -> exp_variable(89);
interference_fringes -> exp_variable(90);
visibility_of_interference_fringes -> exp_variable(91);
interference_fringes -> exp_variable(92);
visibility_of_interference_fringes -> exp_variable(93);
interference_fringes -> exp_variable(94);
visibility_of_interference_fringes -> exp_variable(95);
interference_fringes -> exp_variable(96);
visibility_of_interference_fringes -> exp_variable(97);
interference_fringes -> exp_variable(98);
visibility_of_interference_fringes -> exp_variable(99);
interference_fringes -> exp_variable(100);
visibility_of_interference_fringes -> exp_variable(101);
interference_fringes -> exp_variable(102);
visibility_of_interference_fringes -> exp_variable(103);
interference_fringes -> exp_variable(104);
visibility_of_interference_fringes -> exp_variable(105);
visibility_of_interference_fringes -> exp_variable(106);
substrate_asymmetry_in_peak -> exp_variable(107);
substrate_asymmetry_in_peak -> exp_variable(108);
substrate_asymmetry_in_peak -> exp_variable(109);
substrate_asymmetry_in_peak -> exp_variable(110);
substrate_asymmetry_in_peak -> exp_variable(111);
intensity_of_layer_peak -> exp_variable(137);
intensity_of_layer_peak -> exp_variable(138);
intensity_of_layer_peak -> exp_variable(139);
intensity_of_layer_peak -> exp_variable(140);
intensity_of_layer_peak -> exp_variable(141);
intensity_of_layer_peak -> exp_variable(142);
intensity_of_layer_peak -> exp_variable(143);
intensity_of_layer_peak -> exp_variable(144);
intensity_of_layer_peak -> exp_variable(145);
intensity_of_layer_peak -> exp_variable(146);
layer_asymmetry_in_peak -> exp_variable(112);
layer_asymmetry_in_peak -> exp_variable(113);
layer_asymmetry_in_peak -> exp_variable(114);
layer_asymmetry_in_peak -> exp_variable(115);
layer_asymmetry_in_peak -> exp_variable(116);
layer_wedge_shaped_peak -> exp_variable(117);
layer_wedge_shaped_peak -> exp_variable(118);
layer_wedge_shaped_peak -> exp_variable(119);
layer_wedge_shaped_peak -> exp_variable(120);
layer_wedge_shaped_peak -> exp_variable(121);
layer_wedge_shaped_peak -> exp_variable(122);
layer_wedge_shaped_peak -> exp_variable(123);
layer_wedge_shaped_peak -> exp_variable(124);
layer_wedge_shaped_peak -> exp_variable(125);
layer_wedge_shaped_peak -> exp_variable(126);
evidence_of_mismatch -> exp_variable(127);
evidence_of_mismatch -> exp_variable(128);
evidence_of_mismatch -> exp_variable(129);
evidence_of_mismatch -> exp_variable(130);
evidence_of_mismatch -> exp_variable(131);
substrate_peak_broadening -> exp_variable(132);
substrate_peak_broadening -> exp_variable(133);
substrate_peak_broadening -> exp_variable(134);
substrate_peak_broadening -> exp_variable(135);
substrate_peak_broadening -> exp_variable(136);
intensity_of_layer_peak -> exp_variable(147);
intensity_of_layer_peak -> exp_variable(148);

;;; END of program.
```

[6] The program 'rules3.p' uses the numerical representation to set up the rules for the third partition of the fuzzy rulebase.

```
;;; PROGRAM rules3.p

;;; This program sets up the rules for multiple layers structure
;;; PROGRAM IS CALLED FROM
;;;   OBJECT [FUZZY_SYSTEM] METHOD [load_the_rules_from_file]

sysopen("rno3",0,"line") -> filed;
datafile(filed) -> number_of_rules;
sysclose(filed);

sysopen("rules3",0,"line") -> filed;
datafile(filed) -> existing_rules;
sysclose(filed);

sysopen("exper",0,"line") -> filed;
datafile(filed) -> dummy_array;
sysclose(filed);

sysopen("frules3",0,"line") -> filed;
datafile(filed) -> fuzzy_rules;
sysclose(filed);

sysopen("memfunc3",0,"line") -> filed;
datafile(filed) -> membership_functions;
sysclose(filed);

sysopen("memf3",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);

sysopen("fc3",0,"line") -> filed;
datafile(filed) -> consequences;
sysclose(filed);

sysopen("l3",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);


newarray([1 7]) -> fp;
newarray([1 4]) -> bp;

;;; FUZZY PREMISES
;;; substrate_peak_broadening
dummy_array(15) -> fp(1);
;;; substrate_asymmetry_in_peak
dummy_array(17) -> fp(2);
;;; multiple_layers_asymmetry_in_peak
dummy_array(20) -> fp(3);
;;; multiple_layers_wedge_shaped_peak
dummy_array(22) -> fp(4);
;;; interference_fringes
dummy_array(27) -> fp(5);
;;; visibility_of_interference_fringes
dummy_array(29) -> fp(6);
;;; evidence_of_mismatch
evidence_of_mismatch -> fp(7);

;;; BOOLEAN PREMISES
;;; type_of_structure_is_multiple_layers
dummy_array(30) -> bp(1);
;;; correspondence_between_layers_and_peaks
dummy_array(31) -> bp(2);
;;; number_of_peaks_is_none
dummy_array(32) -> bp(3);
;;; split_substrate_peak
dummy_array(33) -> bp(4);

load rules.p

 newarray([1 "number_of_fuzzy_variables])->exp_variable;
 evidence_of_mismatch -> exp_variable(1);
 evidence_of_mismatch -> exp_variable(2);
 evidence_of_mismatch -> exp_variable(3);
 evidence_of_mismatch -> exp_variable(4);
 evidence_of_mismatch -> exp_variable(5);
 interference_fringes -> exp_variable(6);
 interference_fringes -> exp_variable(7);
 interference_fringes -> exp_variable(8);
 interference_fringes -> exp_variable(9);
 interference_fringes -> exp_variable(10);
 substrate_peak_broadening -> exp_variable(11);
 substrate_peak_broadening -> exp_variable(12);
 substrate_peak_broadening -> exp_variable(13);
 substrate_peak_broadening -> exp_variable(14);
 substrate_peak_broadening -> exp_variable(15);
 interference_fringes -> exp_variable(16);
 interference_fringes -> exp_variable(17);
 interference_fringes -> exp_variable(18);
 interference_fringes -> exp_variable(19);
 interference_fringes -> exp_variable(20);
 interference_fringes -> exp_variable(21);
 interference_fringes -> exp_variable(22);
 interference_fringes -> exp_variable(23);
 interference_fringes -> exp_variable(24);
 interference_fringes -> exp_variable(25);

;;; END of program.
```

[7] The program 'rules4.p' uses the numerical representation to set up the rules for the fourth partition of the fuzzy rulebase.

```
;;; PROGRAM rules4.p

;;; This program sets up the rules for MQW structure and superlattices
;;; PROGRAM IS CALLED FROM
;;;   OBJECT [FUZZY_SYSTEM] METHOD [load_the_rules_from_file]

sysopen("rno4",0,"line") -> filed;
datafile(filed) -> number_of_rules;
sysclose(filed);

sysopen("rules4",0,"line") -> filed;
datafile(filed) -> existing_rules;
sysclose(filed);

sysopen("exper",0,"line") -> filed;
datafile(filed) -> dummy_array;
sysclose(filed);

sysopen("frules4",0,"line") -> filed;
datafile(filed) -> fuzzy_rules;
sysclose(filed);

sysopen("memfunc4",0,"line") -> filed;
datafile(filed) -> membership_functions;
sysclose(filed);

sysopen("memf4",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);

sysopen("fc4",0,"line") -> filed;
datafile(filed) -> consequences;
sysclose(filed);

sysopen("l4",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);


newarray([1 9]) -> fp;
newarray([1 9]) -> bp;

;;; FUZZY PREMISES
;;; satellite_visibility
dummy_array(14) -> fp(1);
;;; satellite_asymmetry_of_plus_minus_peaks
dummy_array(16) -> fp(2);
;;; separation_of_satellite_peaks
separation_of_satellite_peaks -> fp(3);
;;; substrate_peak_broadening
dummy_array(25) -> fp(4);
;;; substrate_asymmetry_in_peak
dummy_array(27) -> fp(5);
;;; zero_order_asymmetry_in_peak
dummy_array(30) -> fp(6);
;;; interference_fringes
dummy_array(31) -> fp(7);
;;; visibility_of_interference_fringes
dummy_array(33) -> fp(8);
;;; evidence_of_mismatch
evidence_of_mismatch ->fp(9);


;;; BOOLEAN VARIABLES
;;; type_of_structure_is_MQW
dummy_array(41) -> bp(1);
;;; satellite_spacing_greater_than_zero
dummy_array(42) -> bp(2);
;;; satellite_subsidiary_interference_effects
dummy_array(21) -> bp(3);
;;; satellite_broadening_of_higher_order_peaks
dummy_array(23) -> bp(4);
;;; satellite_relative_widths_of_peaks_greater_than_zero
dummy_array(43) -> bp(5);
;;; satellite_relative_integrated_intensities_greater_than_zero
dummy_array(44) -> bp(6);
;;; thickness_of_superlattice_greater_than_half_micron
dummy_array(45) -> bp(7);
;;; type_of_structure_has_additional_layers
dummy_array(60) -> bp(8);
;;; number_of_peaks_is_none
dummy_array(46) -> bp(9);

load rules.p

 newarray([1 ^number_of_fuzzy_variables])->exp_variable;
 evidence_of_mismatch -> exp_variable(1);
 evidence_of_mismatch -> exp_variable(2);
 evidence_of_mismatch -> exp_variable(3);
 evidence_of_mismatch -> exp_variable(4);
 evidence_of_mismatch -> exp_variable(5);
 separation_of_satellite_peaks -> exp_variable(6);
 separation_of_satellite_peaks -> exp_variable(7);
 separation_of_satellite_peaks -> exp_variable(8);
 separation_of_satellite_peaks -> exp_variable(9);
 separation_of_satellite_peaks -> exp_variable(10);
 separation_of_satellite_peaks -> exp_variable(11);
 separation_of_satellite_peaks -> exp_variable(12);
 separation_of_satellite_peaks -> exp_variable(13);
 separation_of_satellite_peaks -> exp_variable(14);
 separation_of_satellite_peaks -> exp_variable(15);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(16);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(17);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(18);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(19);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(20);
 satellite_visibility -> exp_variable(21);
 satellite_visibility -> exp_variable(22);
 satellite_visibility -> exp_variable(23);
 satellite_visibility -> exp_variable(24);
 satellite_visibility -> exp_variable(25);
 separation_of_satellite_peaks -> exp_variable(26);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(27);
 separation_of_satellite_peaks -> exp_variable(28);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(29);
 separation_of_satellite_peaks -> exp_variable(30);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(31);
 separation_of_satellite_peaks -> exp_variable(32);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(33);
 separation_of_satellite_peaks -> exp_variable(34);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(35);
 separation_of_satellite_peaks -> exp_variable(36);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(37);
 separation_of_satellite_peaks -> exp_variable(38);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(39);
 separation_of_satellite_peaks -> exp_variable(40);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(41);
 separation_of_satellite_peaks -> exp_variable(42);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(43);
 separation_of_satellite_peaks -> exp_variable(44);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(45);
 separation_of_satellite_peaks -> exp_variable(46);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(47);
 separation_of_satellite_peaks -> exp_variable(48);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(49);
 separation_of_satellite_peaks -> exp_variable(50);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(51);
 separation_of_satellite_peaks -> exp_variable(52);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(53);
 separation_of_satellite_peaks -> exp_variable(54);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(55);
 separation_of_satellite_peaks -> exp_variable(56);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(57);
 separation_of_satellite_peaks -> exp_variable(58);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(59);
 separation_of_satellite_peaks -> exp_variable(60);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(61);
 separation_of_satellite_peaks -> exp_variable(62);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(63);
 separation_of_satellite_peaks -> exp_variable(64);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(65);
 separation_of_satellite_peaks -> exp_variable(66);
 satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(67);
 separation_of_satellite_peaks -> exp_variable(68);
```

```
satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(69);
separation_of_satellite_peaks -> exp_variable(70);
satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(71);
separation_of_satellite_peaks -> exp_variable(72);
satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(73);
separation_of_satellite_peaks -> exp_variable(74);
satellite_asymmetry_of_plus_and_minus_peaks -> exp_variable(75);
satellite_visibility -> exp_variable(76);
satellite_visibility -> exp_variable(77);
satellite_visibility -> exp_variable(78);
satellite_visibility -> exp_variable(79);
satellite_visibility -> exp_variable(80);
interference_fringes -> exp_variable(81);
interference_fringes -> exp_variable(82);
interference_fringes -> exp_variable(83);
interference_fringes -> exp_variable(84);
interference_fringes -> exp_variable(85);
substrate_peak_broadening -> exp_variable(86);
substrate_peak_broadening -> exp_variable(87);
substrate_peak_broadening -> exp_variable(88);
substrate_peak_broadening -> exp_variable(89);
substrate_peak_broadening -> exp_variable(90);

;;; END of program.
```

[8] The program 'rules.p' sets up the rulebase from the numerical representation.

```
;;; PROGRAM rules.p

;;; This program builds the rules (for each structure) from the lists
;;; of rule numbers stored in the array existing_rules.
;;; The ruleset is then stored in the (1x3) array called rule.

;;; This program is called from: rules1.p; rules2.p; rules3.p; or rules4.p
;;; depending on the type of structure that is currently being investigated.

newarray([1 4 1 ^number_of_rules]) -> rule;

;;; ADD THE BOOLEAN PREMISES TO THE RULE

1 -> i;
until i > number_of_rules
do
 [] -> rule_list;

 existing_rules(i,2) -> temp_list;

 while not(temp_list=[])
 do
  1 -> count;
  hd(temp_list) -> value;
  until value < 3
  do
   value - 2 -> value;
   count + 1 -> count
  enduntil;

  rule_list <> [1] -> rule_list;
  bp(count) -> Boolean;
  rule_list <> [^Boolean] -> rule_list;
  rule_list <> [^is_equal_to] -> rule_list;

  if value = 1
  then
   rule_list <> [1] -> rule_list
  else
   rule_list <> [0] -> rule_list
  endif;

  tl(temp_list) -> temp_list;

  if not(temp_list=[])
  then
   rule_list <> [^and] -> rule_list
  endif;
 endwhile;

;;; ADD THE FUZZY PREMISES TO THE RULE

existing_rules(i,1) -> temp_list;

1 -> number_of_variables;
until number_of_variables > 3
do
 if length(temp_list) = number_of_variables
 then
  fuzzy_rules(number_of_variables, 1) -> list1;
  fuzzy_rules(number_of_variables, 2) -> list2;

  newarray([1 ^number_of_variables]) -> dummy2;

  until hd(list1) = i
  do
   1 -> k;
   tl(list1) -> list1;
   until k = number_of_variables
   do
    tl(list2) -> list2;
    k+1 -> k
   enduntil;
  enduntil;

  1 -> k;
  until k > number_of_variables
  do
   hd(list2) -> dummy2(k);
   tl(list2) -> list2;
   k+1 -> k
  enduntil;
 endif;
 number_of_variables + 1 -> number_of_variables
enduntil;

1 -> k;

while not(temp_list = [])
do
 if not(rule_list=[])
 then
  rule_list <> [^and] -> rule_list
 endif;

 1 -> count;
 hd(temp_list) -> value;
 until value < 6
 do
  value - 5 -> value;
  count + 1 -> count
 enduntil;

 rule_list <> [2] -> rule_list;
 fp(count) -> fuzzy;
 rule_list <> [^fuzzy] -> rule_list;

 rule_list <> [^is_equal_to] -> rule_list;

 if value = 1
 then
  rule_list <> ['NONE'] -> rule_list
 elseif value = 2
 then
  rule_list <> ['SOME'] -> rule_list
 elseif value = 3
 then
  rule_list <> ['FAIRLY'] -> rule_list
 elseif value = 4
 then
  rule_list <> ['VERY'] -> rule_list
 else
  rule_list <> ['EXTREME'] -> rule_list
 endif;

 dummy2(k) -> mem;
 mf(mem) -> mem;
 membership_functions(mem) -> membership;

 rule_list <> [^membership] -> rule_list;
 k+1 -> k;
 tl(temp_list) -> temp_list
endwhile;

rule_list -> rule(1,i);

;;; ADD THE FUZZY CONSEQUENTS TO THE RULE

[] -> rule_list;

existing_rules(i,4) -> temp_list;
hd(temp_list) -> count;
rule_list <> [^count] -> rule_list;

consequences(count) -> fuzzy;
rule_list <> [^fuzzy] -> rule_list;

rule_list <> [^is_equal_to] -> rule_list;

existing_rules(i,3) -> temp_list;
hd(temp_list) -> value;

if value = 1
then
 rule_list <> ['NONE'] -> rule_list
elseif value = 2
then
 rule_list <> ['SOME'] -> rule_list
elseif value = 3
then
```

```
    rule_list < > ['FAIRLY'] -> rule_list
  elseif value = 4
  then
    rule_list < > ['VERY'] -> rule_list
  else
    rule_list < > ['EXTREME'] -> rule_list
  endif;

  rule_list -> rule(2,i);


  ;;;  ADD THE LAYER NUMBER TO THE RULE

  layers(i) -> value;
  [] -> rule_list;
  rule_list < > [^value] -> rule_list;

  rule_list -> rule(3,i);

  i+1 -> i;

enduntil;

;;;  END of program.
```

[9] The program 'fuzzout.p' is used to calculate defuzzified output values and stores these values in variables.

```
;;;          PROGRAM: fuzzout.p
;;;
;;;          THIS PROGRAM STORES FUZZY OUTPUT VARIABLES
;;;

if type_of_structure = 'substrate only'
then
        sysopen("fuzzno1",0,"line") -> filed;
        datafile(filed) -> number_of_fuzzy_variables;
        sysclose(filed);

        1 -> i;
        until i > 7
        do
                make_instance([FUZZY_VARIABLE]) -> B;
                B <- calculate_value_of_conclusion(i);
                ;;; This method stores the calculated value in
                ;;; fuzzy_variable(i,6)
                i+1 -> i;
        enduntil;

        fuzzy_variable(1,6) -> crystal_quality;
        fuzzy_variable(2,6) -> strain_in_surface_layer_of_sample;
        fuzzy_variable(3,6)
                -> reference_crystal_is_different_to_substrate;
        fuzzy_variable(4,6) -> crystal_consists_of_subgrains;
        fuzzy_variable(5,6) -> characteristic_curve;
        fuzzy_variable(6,6) -> misorientation_of_substrate;
        fuzzy_variable(7,6) -> incorrect_experiment;

elseif type_of_structure = 'a single layer'
then
        sysopen("fuzzno2",0,"line") -> filed;
        datafile(filed) -> number_of_fuzzy_variables;
        sysclose(filed);

        1 -> i;
        until i > 18
        do
                make_instance([FUZZY_VARIABLE]) -> B;
                B <- calculate_value_of_conclusion(i);
                i+1 -> i;
        enduntil;

        fuzzy_variable(1,6) -> crystal_quality;
        fuzzy_variable(2,6) -> characteristic_curve;
        fuzzy_variable(3,6) -> misorientation_of_substrate;
        fuzzy_variable(4,6) -> bending_of_substrate;
        fuzzy_variable(5,6) -> grading_of_the_layer;
        fuzzy_variable(6,6) -> layer_is_thick;
        fuzzy_variable(7,6) -> change_in_lattice_parameter_with_depth;
        fuzzy_variable(8,6) -> layer_is_present_in_the_substrate_peak;
        fuzzy_variable(9,6) -> layer_is_thin;
        fuzzy_variable(10,6) -> evidence_of_mismatch;
        fuzzy_variable(11,6) -> peak_is_outside_the_scan_range;
        fuzzy_variable(12,6) -> misoriented_or_mismatched_layer;
        fuzzy_variable(13,6) -> multiple_layers;
        fuzzy_variable(14,6)
                -> simulation_or_calibration_chart_is_needed;
        fuzzy_variable(15,6) -> relaxation;
        fuzzy_variable(16,6) -> incorrect_experiment;
        fuzzy_variable(17,6) -> thickness_of_layer;
        fuzzy_variable(18,6) -> experimental_mismatch;

elseif type_of_structure = 'multiple layers'
then
        sysopen("fuzzno3",0,"line") -> filed;
        datafile(filed) -> number_of_fuzzy_variables;
        sysclose(filed);

        1 -> i;
        until i > 10
        do
                make_instance([FUZZY_VARIABLE]) -> B;
                B <- calculate_value_of_conclusion(i);
                i+1 -> i;
        enduntil;

        fuzzy_variable(1,6) -> crystal_quality;
        fuzzy_variable(2,6) -> misorientation_of_substrate;
        fuzzy_variable(3,6) ->simulation_or_calibration_chart_is_needed;
        fuzzy_variable(4,6) -> relaxation;
        fuzzy_variable(5,6) -> layers_are_thick;
        fuzzy_variable(6,6) -> there_are_hidden_layers_somewhere;
        fuzzy_variable(7,6) -> incorrect_experiment;
        fuzzy_variable(8,6) -> evidence_of_mismatch;
        fuzzy_variable(9,6) -> there_are_thin_layers_at_the_interfaces;
        fuzzy_variable(10,6) -> evidence_of_interferometer_structure;
else
        sysopen("fuzzno4",0,"line") -> filed;
        datafile(filed) -> number_of_fuzzy_variables;
        sysclose(filed);

        1 -> i;
        until i > 14
        do
                make_instance([FUZZY_VARIABLE]) -> B;
                B <- calculate_value_of_conclusion(i);
                i+1 -> i;
        enduntil;

        fuzzy_variable(1,6) -> crystal_quality;
        fuzzy_variable(2,6) -> characteristic_curve;
        fuzzy_variable(3,6) -> misorientation_of_substrate;
        fuzzy_variable(4,6) ->simulation_or_calibration_chart_is_needed;
        fuzzy_variable(5,6) -> relaxation;
        fuzzy_variable(6,6) -> layers_are_thick;
        fuzzy_variable(7,6) -> layers_are_thin;
        fuzzy_variable(8,6)
                -> grading_or_dispersion_of_layer_thicknesses;
        fuzzy_variable(9,6) -> grading_occurs_through_AB_layers;
        fuzzy_variable(10,6) -> large_overall_layer_thickness;
        fuzzy_variable(11,6) -> layers_are_not_uniform;
        fuzzy_variable(12,6) -> incorrect_experiment;
        fuzzy_variable(13,6) -> period_of_superlattice;
        fuzzy_variable(14,6) -> period_dispersion;
endif;

;;;          END OF PROGRAM
;;;
```

[10] The program 'thick.p' uses the integrated intensities to calculate an estimate of layer thickness. This is used when there are no interference fringes.

```
;;; PROGRAM: thick.p

;;; Use the integrated intensities to work out an estimate
;;; layer thickness for single layered structures when there
;;; are no interference fringes visible.
;;;
;;; PROGRAM IS CALLED FROM
;;;   OBJECT [STRUCTURAL_PARAMETERS_FRAME]
;;;          METHOD [fill_the_frame]


;;; ____________________________________________________
;;;
;;; 1=Si  2=Ge  3=AlP  4=AlAs  5=AlSb  6=GaP
;;; 7=GaAs  8=GaSb  9=InP  10=InAs  11=InSb
;;;
;;; 12=GaSi    13=InGaP  14=InGaAs
;;; 15=InGaSb  16=AlGaAs  17=InAsP
;;; 18=GaAsP  19=InGaAlAs  20=GaInAsP
;;; ____________________________________________________

;;; Values for thicknesses up to 3 microns:
;;; (1) = 0.5  (2) = 1    (3) = 1.25
;;; (4) = 1.5  (5) = 1.75  (6) = 2
;;; (7) = 2.25  (8) = 2.5  (9) = 2.75
;;; (10) = 3
;;; ____________________________________________________


newarray([1 11 1 20]) -> thickness;

1 -> i;
until i > 11
do
  1 -> j;
  until j > 20
  do
    [0 0 0 0 0 0 0 0 0 0] -> thickness(i,j);
    j+1 -> j;
  enduntil;
  i+1 -> i;
enduntil;

;;; ____________________________________________________


;;; SUBSTRATE: GaAs   LAYER: AlGaAs

[0.275 0.508 0.659 0.768 0.861 0.948 0.977 1.091 1.159 1.214]
                -> thickness(7,16);

;;; SUBSTRATE: GaAs   LAYER: AlGaAs

[0.561 1.12 1.42 1.65 1.87 2.08 2.28 2.48 2.65 2.84] -> thickness(9,14);

;;;____________________________________________________


;;; Calculate the integrated intensity of layer as function of substrate
if layer_integrated_intensity_of_peak = 0
then
  substrate_integrated_intensity_of_peak -> x;
else
  substrate_integrated_intensity_of_peak /
                  layer_integrated_intensity_of_peak -> x;
endif;

;;; Check this value against layer thickness
thickness(mat1,mat2) -> y;

;;; ____________________________________________________

;;; BECAUSE ALL VARIABLES MUST BE DECLARED
;;;              GLOBALLY IN THE PARENT PROGRAM
;;; I AM REUSING VARIABLES IN THE FOLLOWING SECTION OF CODE

;;; This may make the code very difficult to read ...

if (x < y(10) or x = y(10))
then
  ;;; first value to check is zero
  0 -> mat1;
  ;;; work through list of values taken from the array thickness
  1 -> i;
  until i > 10
  do
    ;;; take first value from the list
    y(i) -> mat2;
    ;;; if x is between mat1 and mat2 then stop
    if (x > mat1 or x = mat1) and (x < mat2 or x = mat2)
    then
      ;;; this stops the loop
      11 -> i;
      ;;; number of the list element is recorded in the variable j
      i -> j;
    else
      ;;; otherwise continue searching through the list of values
      ;;; storing the last value in mat1 and the next value in mat2
      i+1 -> i;
      mat2 -> mat1;
    endif;
  enduntil;

  ;;; find out the difference between the two values taken from the list
  ;;; and store this difference in y
  mat2 - mat1 -> y;
  ;;; find out the difference between the experimental value and the lower
  ;;; value taken from the list
  ;;; store this difference in x
  x - mat1 -> x;
  ;;; Find the second difference as a percentage of the first
  ;;; Store this percentage in x (1)
  (x * 100) / y -> x;

  ;;; Store list of layer thicknesses in the variable y
  [0 0.5 1 1.25 1.5 1.75 2.0 2.25 2.5 2.75 3.0] -> y;

  ;;; Choose the lower value in the interval of layer thicknesses
  ;;; where the experimental value exists
  j-1 -> j;
  ;;; pick this value from the list of layer thicknesses
  y(j) -> i;
  ;;; Increment the j counter to move to the higher value in the interval
  j+1 -> j;
  ;;; Store this layer thickness in the variable j
  y(j) -> j;
  ;;; Find the difference between these two layer thicknesses
  ;;; and store the result in y
  j - i -> y;
  ;;; Find the x (see (1) above) percentage of the value y
  ;;; and store the result in x
  (y * x) / 100 -> x;

  ;;; The lower value for layer thickness plus the percentage of layer
  ;;; thickness gives an extimate of the layer thickness based on the
  ;;; integrated intensities of layer and substrate
  i + x -> thickness_of_layer;

else
  0 -> thickness_of_layer;
endif;

;;;____________________________________________________
```

[11] The program 'induce.p' performs inductive learning of new fuzzy rules from examples.

```
;;; PROGRAM NAME: induce.p

;;; Program for inductive learning from examples

;;; INDEX OF OBJECTS
;;;
;;; 1. Some functions
;;; 2. Function to add rules to a connection matrix
;;; 3. Connection-Matrix Object
;;; 4. Fuzzy-System Object


;;; [1] Some functions

define ismember(element,list);
 if list matches [== ^element ==]
  then true
  else false
 endif
enddefine;

define subset_of(x,y);
 lvars x y check value;

 1 -> check;
 until x=[]
 do
  hd(x) -> value;
  if not(ismember(value,y))
  then
   0 -> check
  endif;
  tl(x) -> x
 enduntil;

 if check=1
  then true
  else false
 endif
enddefine;


;;; [2] FUNCTION TO ADD RULES TO A CONNECTION MATRIX
;;;     (This function uses recursion)

define add_rules_to_CM(matrix,list_of_rules);
 lvars rule_number row column;

 if list_of_rules=[]
 then
 else
  hd(list_of_rules) -> rule_number;
  (rule_number * 2)-1 -> row;
  (rule_number * 2) -> column;

  1 -> matrix(row,column);
  add_rules_to_CM(matrix,tl(list_of_rules))
 endif;
enddefine;

;;; [3] CONNECTION_MATRIX OBJECT

flavour CONNECTION_MATRIX;
 ivars matrix_name matrix list_of_rules credibility_weight;
 ivars history_of_decisions history_of_incs_decs;

 ;;; LIST OF METHODS
 ;;;
 ;;; 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; 2) save_to_file(x);

 ;;; METHOD: 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; Method to create a connection matrix
 defmethod create_a_connection_matrix(number_of_rules rules);
  lvars number_of_rules rules size_of_matrix;

  rules -> list_of_rules;

  number_of_rules * 2 -> size_of_matrix;

  newarray([1 ^size_of_matrix 1 ^size_of_matrix],0) -> matrix;

  ;;; Add rules to the Connection Matrix (uses recursion)
  add_rules_to_CM(matrix,rules)
 enddefmethod;

;;; METHOD 2: save connection matrices to file
defmethod save_to_file(expert, structure);
 lvars expert structure filed;

 if expert = 1
 then
  if structure = 1
  then
   syscreate("me1s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("me1s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("me1s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("me1s4",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  endif;
 elseif expert = 2
 then
  if structure = 1
  then
   syscreate("me2s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("me2s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("me2s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("me2s4",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  endif;
 elseif expert = 3
 then
  if structure = 1
  then
   syscreate("me3s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("me3s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("me3s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("me3s4",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
```

```
      endif;
    elseif expert = 4
     if structure = 1
     then
       syscreate("mc4s1",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif structure = 2
     then
       syscreate("mc4s2",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif structure = 3
     then
       syscreate("mc4s3",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     else
       syscreate("mc4s4",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     endif;
    else
     if structure = 1
     then
       syscreate("mc5s1",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif structure = 2
     then
       syscreate("mc5s2",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     elseif structure = 3
     then
       syscreate("mc5s3",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     else
       syscreate("mc5s4",1,"line") -> filed;
       matrix -> datafile(filed);
       sysclose(filed);
     endif;
    endif;
  enddefmethod;

endflavour;


;;; [4] FUZZY_SYSTEM OBJECT

;;;
;;;
;;;  LIST OF METHODS:
;;;
;;;    1) create_new_rules_by_induction(minimum_sample_size,
;;;                    number_of_occurrences);
;;;    2) create_consequent_membership_functions();
;;;    3) build_an_induction_matrix(list_of_consequents);
;;;    4) calculate_values(value, A);
;;;    5) get_lists_of_potential_rules(list_of_consequents);
;;;    6) check_if_potential_rules_already_exists(i);
;;;    7) check_for_coverage(quota, high_cardinality);
;;;    8) display_new_rules(i);
;;;    9) add_rules_to_rulebase;

flavour FUZZY_SYSTEM;
 ivars type_of_structure fuzzy_variable thresholds number_of_potential_rules;
 ivars consequent_tally example_set_fuzzy_premise;
 ivars example_set_boolean_premise example_set_consequent;
 ivars number_of_fuzzy_premises number_of_boolean_premises;
 ivarsnumber_of_consequencesfuzzy_premisesboolean_premisesconsequences;
 ivars fuzzy_peak_broadening fuzzy_asymmetry;
 ivars fuzzy_wedge_shaped fuzzy_visibility;
 ivarsfuzzy_spacing_of_peaksfuzzy_plus_minus_asymmetryfuzzy_interference;
 ivars fuzzy_intensity_of_peak fuzzy_mismatch membership_functions;
 ivars potential_rules percent1 percent2 neg effectiveness;

 ;;; METHOD 1) create_new_rules_by_induction(minimum_sample_size,
 ;;;                    number_of_occurrences);
 ;;;
 ;;; This method measures the instance values against the consequences and
 ;;; stores this information in a matrix
 defmethod create_new_rules_by_induction(minimum_sample_size,
                    number_of_occurrences);
 lvars minimum_sample_size number_of_occurrences;
 lvars i filed size_of_sample temp_list j k quota consequents_in_rules;

 nl(10);
 pr('    FUZZY SYSTEM FOR X-RAY ROCKING CURVE ANALYSIS');
 nl(3);
 pr('_____________________________________________________');
 nl(2);
 pr('INDUCTIVE LEARNING OF NEW RULES FROM EXAMPLES');
 nl(2);
 ;;; Assign membership functions to variables
 [0.0 0.3 0.1 0.4 0.3 0.6 0.5 0.8 0.7 1.0] -> fuzzy_peak_broadening;
 [0.0 0.1 0.0 0.2 0.1 0.5 0.4 0.8 0.7 1.0] -> fuzzy_asymmetry;
 [0.0 0.1 0.0 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_wedge_shaped;
 [0.0 0.1 0.0 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_visibility;
 [0.0 0.1 0.0 0.3 0.2 0.5 0.4 0.7 0.6 1.0] -> fuzzy_spacing_of_peaks;
 [0.0 0.1 0.0 0.2 0.1 0.5 0.4 0.8 0.7 1.0] -> fuzzy_plus_minus_asymmetry;
 [0.0 0.1 0.0 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_interference;
 [0.0 0.1 0.0 0.2 0.1 0.4 0.3 0.7 0.6 1.0] -> fuzzy_intensity_of_peak;

 1 -> i;
 until i > 4
 do
  if i=1
  then
    pr('Survey substrate only data');nl(2);
    'substrate only' -> type_of_structure;
    sysopen("conrec1",0,"line") -> filed;
    datafile(filed) -> consequent_tally;
    sysclose(filed);
    sysopen("pexset1",0,"line") -> filed;
    datafile(filed) -> example_set_fuzzy_premise;
    sysclose(filed);
    sysopen("bexset1",0,"line") -> filed;
    datafile(filed) -> example_set_boolean_premise;
    sysclose(filed);
    sysopen("cexset1",0,"line") -> filed;
    datafile(filed) -> example_set_consequent;
    sysclose(filed);
    sysopen("cinr1",0,"line") -> filed;
    datafile(filed) -> consequents_in_rules;
    sysclose(filed);

    length(example_set_fuzzy_premise) -> size_of_sample;
    4 -> number_of_fuzzy_premises;
    4 -> number_of_boolean_premises;
    7 -> number_of_consequences;

    newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
    "substrate_peak_broadening" -> fuzzy_premises(1);
    "substrate_asymmetry_in_peak" -> fuzzy_premises(2);
    "interference_fringes" -> fuzzy_premises(3);
    "visibility_of_interference_fringes" -> fuzzy_premises(4);

    newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
    "type_of_structure_is_substrate_only" -> boolean_premises(1);
    "number_of_peaks_is_one" -> boolean_premises(2);
    "number_of_peaks_is_more_than_one" -> boolean_premises(3);
    "number_of_peaks_is_none" -> boolean_premises(4);

    newarray([1 ^number_of_consequences]) -> consequences;
    "crystal_quality" -> consequences(1);
    "strain_in_surface_layer_of_sample" -> consequences(2);
    "reference_crystal_is_different_to_substrate" -> consequences(3);
    "crystal_consists_of_subgrains" -> consequences(4);
    "characteristic_curve" -> consequences(5);
    "misorientation_of_substrate" -> consequences(6);
    "incorrect_experiment" -> consequences(7);

    newarray([1 4]) -> membership_functions;
    fuzzy_peak_broadening -> membership_functions(1);
    fuzzy_asymmetry -> membership_functions(2);
    fuzzy_interference -> membership_functions(3);
    fuzzy_visibility -> membership_functions(4);
  elseif i=2
  then
    nl(2);
    pr('Survey single layer data');nl(2);
    'single layer' -> type_of_structure;
    sysopen("conrec2",0,"line") -> filed;
    datafile(filed) -> consequent_tally;
    sysclose(filed);
    sysopen("bexset2",0,"line") -> filed;
```

```
 datafile(filed) -> example_set_boolean_premise;
 sysclose(filed);
 sysopen("pexset2",0,"line") -> filed;
 datafile(filed) -> example_set_fuzzy_premise;
 sysclose(filed);
 sysopen("cexset2",0,"line") -> filed;
 datafile(filed) -> example_set_consequent;
 sysclose(filed);
 sysopen("cinr2",0,"line") -> filed;
 datafile(filed) -> consequents_in_rules;
 sysclose(filed);

 length(example_set_fuzzy_premise) -> size_of_sample;
 7 -> number_of_fuzzy_premises;
 16 -> number_of_boolean_premises;
 18 -> number_of_consequences;
 newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
 "substrate_peak_broadening" -> fuzzy_premises(1);
 "substrate_asymmetry_in_peak" -> fuzzy_premises(2);
 "layer_asymmetry_in_peak" -> fuzzy_premises(3);
 "layer_wedge_shaped_peak" -> fuzzy_premises(4);
 "interference_fringes" -> fuzzy_premises(5);
 "visibility_of_interference_fringes" -> fuzzy_premises(6);
 "intensity_of_layer_peak" -> fuzzy_premises(7);

 newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
 "type_of_structure_is_single_layer" -> boolean_premises(1);
 "number_of_peaks_is_one" -> boolean_premises(2);
 "number_of_peaks_is_two" -> boolean_premises(3);
 "number_of_peaks_is_more_than_two" -> boolean_premises(4);
 "number_of_peaks_is_none" -> boolean_premises(5);
 "substrate_material_equal_to_layer" -> boolean_premises(6);
 "peak_separation_is_low" -> boolean_premises(7);
 "layer_split_peak" -> boolean_premises(8);
 "layer_integrated_intensity_of_peak_is_zero" -> boolean_premises(9);
 "layer_thickness_greater_than_half_micron" -> boolean_premises(10);
 "layer_thickness_less_than_5_microns" -> boolean_premises(11);
 "peak_splitting_is_zero" -> boolean_premises(12);
 "peak_splitting_less_than_three_times_width_of_peak"
              -> boolean_premises(13);
 "relaxed_mismatch_is_high" -> boolean_premises(14);
 "peak_splitting_is_high" -> boolean_premises(15);
 "spacing_of_interference_fringes_is_low" -> boolean_premises(16);

 newarray([1 ^number_of_consequences]) -> consequences;
 "crystal_quality" -> consequences(1);
 "characteristic_curve" -> consequences(2);
 "misorientation_of_substrate" -> consequences(3);
 "bending_of_substrate" -> consequences(4);
 "grading_of_the_layer" -> consequences(5);
 "layer_is_thick" -> consequences(6);
 "change_in_lattice_parameter_with_depth" -> consequences(7);
 "layer_is_present_in_the_substrate_peak" -> consequences(8);
 "layer_is_thin" -> consequences(9);
 "evidence_of_mismatch" -> consequences(10);
 "peak_is_outside_the_scan_range" -> consequences(11);
 "misoriented_or_mismatched_layer" -> consequences(12);
 "multiple_layers" -> consequences(13);
 "simulation_or_calibration_chart_is_needed" -> consequences(14);
 "relaxation" -> consequences(15);
 "incorrect_experiment" -> consequences(16);
 "thickness_of_layer" -> consequences(17);
 "experimental_mismatch" -> consequences(18);

 newarray([1 7]) -> membership_functions;
 fuzzy_peak_broadening -> membership_functions(1);
 fuzzy_asymmetry -> membership_functions(2);
 fuzzy_asymmetry -> membership_functions(3);
 fuzzy_wedge_shaped -> membership_functions(4);
 fuzzy_interference -> membership_functions(5);
 fuzzy_visibility -> membership_functions(6);
 fuzzy_intensity_of_peak -> membership_functions(7);
elseif i=3
then
 nl(2);
 pr('Survey multiple layers data');nl(2);
 'multiple layers' -> type_of_structure;
 sysopen("conrec3",0,"line") -> filed;
 datafile(filed) -> consequent_tally;
 sysclose(filed);
 sysopen("pexset3",0,"line") -> filed;
 datafile(filed) -> example_set_fuzzy_premise;
 sysclose(filed);
 sysopen("bexset3",0,"line") -> filed;
 datafile(filed) -> example_set_boolean_premise;
 sysclose(filed);
 sysopen("cexset3",0,"line") -> filed;
 datafile(filed) -> example_set_consequent;
 sysclose(filed);
 sysopen("cinr3",0,"line") -> filed;
 datafile(filed) -> consequents_in_rules;
 sysclose(filed);

 length(example_set_fuzzy_premise) -> size_of_sample;
 6 -> number_of_fuzzy_premises;
 4 -> number_of_boolean_premises;
 10 -> number_of_consequences;
 newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
 "substrate_peak_broadening" -> fuzzy_premises(1);
 "substrate_asymmetry_in_peak" -> fuzzy_premises(2);
 "multiple_layers_asymmetry_in_peak" -> fuzzy_premises(3);
 "multiple_layers_wedge_shaped_peak" -> fuzzy_premises(4);
 "interference_fringes" -> fuzzy_premises(5);
 "visibility_of_interference_fringes" -> fuzzy_premises(6);

 newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
 "type_of_structure_is_multiple_layers" -> boolean_premises(1);
 "correspondence_between_layers_and_peaks" -> boolean_premises(2);
 "number_of_peaks_is_none" -> boolean_premises(3);
 "split_substrate_peak" -> boolean_premises(4);

 newarray([1 ^number_of_consequences]) -> consequences;
 "crystal_quality" -> consequences(1);
 "misorientation_of_substrate" -> consequences(2);
 "simulation_or_calibration_chart_is_needed" -> consequences(3);
 "relaxation" -> consequences(4);
 "layers_are_thick" -> consequences(5);
 "there_are_hidden_layers_somewhere" -> consequences(6);
 "incorrect_experiment" -> consequences(7);
 "evidence_of_mismatch" -> consequences(8);
 "there_are_thin_layers_at_the_interfaces" -> consequences(9);
 "evidence_of_interferometer_structure" -> consequences(10);

 newarray([1 6]) -> membership_functions;
 fuzzy_peak_broadening -> membership_functions(1);
 fuzzy_asymmetry -> membership_functions(2);
 fuzzy_asymmetry -> membership_functions(3);
 fuzzy_wedge_shaped -> membership_functions(4);
 fuzzy_interference -> membership_functions(5);
 fuzzy_visibility -> membership_functions(6);

elseif i=4
then
 nl(2);
 pr('Survey MQW and Superlattice data');nl(2);
 'MQW structure' -> type_of_structure;
 sysopen("conrec4",0,"line") -> filed;
 datafile(filed) -> consequent_tally;
 sysclose(filed);
 sysopen("pexset4",0,"line") -> filed;
 datafile(filed) -> example_set_fuzzy_premise;
 sysclose(filed);
 sysopen("bexset4",0,"line") -> filed;
 datafile(filed) -> example_set_boolean_premise;
 sysclose(filed);
 sysopen("cexset4",0,"line") -> filed;
 datafile(filed) -> example_set_consequent;
 sysclose(filed);
 sysopen("cinr4",0,"line") -> filed;
 datafile(filed) -> consequents_in_rules;
 sysclose(filed);

 length(example_set_fuzzy_premise) -> size_of_sample;
 8 -> number_of_fuzzy_premises;
 9 -> number_of_boolean_premises;
 14 -> number_of_consequences;

 newarray([1 ^number_of_fuzzy_premises]) -> fuzzy_premises;
 "satellite_visibility" -> fuzzy_premises(1);
 "satellite_asymmetry_of_plus_minus_peaks" -> fuzzy_premises(2);
 "separation_of_satellite_peaks" -> fuzzy_premises(3);
 "substrate_peak_broadening" -> fuzzy_premises(4);
 "substrate_asymmetry_in_peak" -> fuzzy_premises(5);
 "zero_order_asymmetry_in_peak" -> fuzzy_premises(6);
 "interference_fringes" -> fuzzy_premises(7);
 "visibility_of_interference_fringes" -> fuzzy_premises(8);

 newarray([1 ^number_of_boolean_premises]) -> boolean_premises;
```

```
  "type_of_structure_is_MQW" -> boolean_premises(1);
  "satellite_spacing_greater_than_zero" -> boolean_premises(2);
  "satellite_subsidiary_interference_effects" -> boolean_premises(3);
  "satellite_broadening_of_higher_order_peaks" -> boolean_premises(4);
  "satellite_relative_widths_of_peaks_greater_than_zero"
                -> boolean_premises(5);
  "satellite_relative_integrated_intensities_greater_than_zero"
                -> boolean_premises(6);
  "thickness_of_superlattice_greater_than_half_micron"
                -> boolean_premises(7);
  "type_of_structure_has_additional_layers" -> boolean_premises(8);
  "number_of_peaks_is_none" -> boolean_premises(9);

  newarray([1 ^number_of_consequences]) -> consequences;
  "crystal_quality" -> consequences(1);
  "characteristic_curve" -> consequences(2);
  "misorientation_of_substrate" -> consequences(3);
  "simulation_or_calibration_chart_is_needed" -> consequences(4);
  "relaxation" -> consequences(5);
  "layers_are_thick" -> consequences(6);
  "layers_are_thin" -> consequences(7);
  "grading_or_dispersion_of_layer_thicknesses" -> consequences(8);
  "grading_occurs_through_AB_layers" -> consequences(9);
  "large_overall_layer_thickness" -> consequences(10);
  "layers_are_not_uniform" -> consequences(11);
  "incorrect_experiment" -> consequences(12);
  "period_of_superlattice" -> consequences(13);
  "period_dispersion" -> consequences(14);

  newarray([1 8]) -> membership_functions;
  fuzzy_visibility -> membership_functions(1);
  fuzzy_plus_minus_asymmetry -> membership_functions(2);
  fuzzy_spacing_of_peaks -> membership_functions(3);
  fuzzy_peak_broadening -> membership_functions(4);
  fuzzy_asymmetry -> membership_functions(5);
  fuzzy_asymmetry -> membership_functions(6);
  fuzzy_interference -> membership_functions(7);
  fuzzy_visibility -> membership_functions(8)
else
endif;

;;; Quota of rules for effectiveness fitness function
1 -> quota;

newarray([1 ^number_of_consequences]) -> effectiveness;
1 -> k;
until k > number_of_consequences
do
  if consequents_in_rules(k) < quota
  then
    1 -> effectiveness(k)
  else
    0 -> effectiveness(k)
  endif;
  k + 1 -> k
enduntil;

;;; METHOD 2:
pr('  creating membership functions for the consequents');nl(1);
^create_consequent_membership_functions;

if (size_of_sample > minimum_sample_size or
                size_of_sample = minimum_sample_size)
then
  pr('  checking sample size for each consequent');nl(1);
  [] -> temp_list;

  1 -> j;
  until j > number_of_consequences
  do
    if consequent_tally(j) > number_of_occurrences
    then
      temp_list <> [^j] -> temp_list
    endif;
    j + 1 -> j
  enduntil;

  0 -> number_of_potential_rules;

  if not(temp_list = [])
  then
    10 -> number_of_potential_rules;
    0.95 -> thresholds;
    0.25 -> neg;
    until number_of_potential_rules > 10
    do
      nl(1);
      pr('  positive threshold = ');pr(thresholds);nl(1);
      pr('  negative threshold = ');pr(neg);nl(1);
      if thresholds = 0.9
      then
        11 -> number_of_potential_rules
      endif;
      ;;;  METHOD 3:
      pr('  building the PIT and NIT matrices');nl(1);
      ^build_an_induction_matrix(temp_list);
      ;;;  METHOD 5:
      pr('  checking for potential rules in the induction test matrices');nl(1);
      ^get_lists_of_potential_rules(temp_list);
      thresholds - 0.05 -> thresholds;
      neg + 0.025 -> neg;
    enduntil;
  endif;
else
  [] -> temp_list;
endif;
nl(1);
pr('  Check fitness function for CORRECTNESS');nl(1);
pr('  Check fitness function for EFFECTIVENESS');nl(1);

;;;  METHOD 6:

pr('  Check if any of the potential rules already exist');nl(1);
pr('  Check fitness function for SIMPLICITY');nl(1);
if not(temp_list=[])
then
  ^check_if_potential_rules_already_exists(i);
endif;

;;;  METHOD 7:
pr('  Check fitness function for COVERAGE');nl(1);
pr('  Check fitness function for ROBUSTNESS');nl(1);
if not(temp_list=[])
then
  ^check_for_coverage(10, 50);
endif;

if not(temp_list=[])
then
  nl(3);
  pr('_____________________________________________________');
  nl(2);
  if i=1
  then
    pr('  SUBSTRATE ONLY STRUCTURE :');nl(2);
  elseif i=2
  then
    pr('  SINGLE LAYER STRUCTURE :');nl(2);
  elseif i=3
  then
    pr('  MULTIPLE LAYERS STRUCTURE :');nl(2);
  else
    pr('  MQWs AND SUPERLATTICES :');nl(2);
  endif;
  pr('   NEW FUZZY RULES CREATED BY
                INDUCTIVE LEARNING');
  nl(2);
  pr('   THESE RULES HAVE BEEN VALIDATED BY
                FITNESS FUNCTIONS');
  nl(1);
  ^display_new_rules(i);
  pr('_____________________________________________________');
  nl(2);
endif;

 i + 1 -> i;
enduntil;

^add_rules_to_rulebase;

enddefmethod;
```

```
;;; METHOD 2) create_consequent_membership_functions();
;;;
;;; This method creates the membership functions for the consequents
defmethod create_consequent_membership_functions;

 if type_of_structure = 'substrate only'
 then

  newarray([1 7]) -> fuzzy_variable;

  ;;; crystal_quality;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1);
  ;;; strain_in_surface_layer_of_sample;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2);
  ;;; reference_crystal_is_different_to_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3);
  ;;; crystal_consists_of_subgrains;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4);
  ;;; characteristic_curve;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5);
  ;;; misorientation_of_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6);
  ;;; incorrect_experiment;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7);

 elseif type_of_structure = 'single layer'
 then

  newarray([1 18]) -> fuzzy_variable;

  ;;; crystal_quality;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1);
  ;;; characteristic_curve;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2);
  ;;; misorientation_of_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3);
  ;;; bending_of_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4);
  ;;; grading_of_the_layer;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5);
  ;;; layer_is_thick;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6);
  ;;; change_in_lattice_parameter_with_depth;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7);
  ;;; layer_is_present_in_the_substrate_peak;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8);
  ;;; layer_is_thin;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9);
  ;;; evidence_of_mismatch;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10);
  ;;; peak_is_outside_the_scan_range;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(11);
  ;;; misoriented_or_mismatched_layer;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(12);
  ;;; multiple_layers;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(13);
  ;;; simulation_or_calibration_chart_is_needed;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(14);
  ;;; relaxation;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(15);
  ;;; incorrect_experiment;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(16);
  ;;; thickness_of_layer;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(17);
  ;;; experimental_mismatch;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(18);

 elseif type_of_structure = 'multiple layers'
 then

  newarray([1 10]) -> fuzzy_variable;

  ;;; crystal_quality;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1);
  ;;; misorientation_of_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2);
  ;;; simulation_or_calibration_chart_is_needed;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3);
  ;;; relaxation;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4);
  ;;; layers_are_thick;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5);
  ;;; there_are_hidden_layers_somewhere;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6);
  ;;; evidence_of_mismatch
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7);
  ;;; incorrect_experiment;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8);
  ;;; interferometer_structure
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9);
  ;;; there_are_thin_layers_at_the_interfaces
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10);

 else
  newarray([1 14]) -> fuzzy_variable;

  ;;; crystal_quality;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(1);
  ;;; characteristic_curve;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(2);
  ;;; misorientation_of_substrate;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(3);
  ;;; simulation_or_calibration_chart_is_needed;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(4);
  ;;; relaxation;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(5);
  ;;; layers_are_thick;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(6);
  ;;; layers_are_thin;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(7);
  ;;; grading_or_dispersion_of_layer_thicknesses;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(8);
  ;;; grading_occurs_through_AB_layers;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(9);
  ;;; large_overall_layer_thickness;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(10);
  ;;; layers_are_not_uniform;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(11);
  ;;; incorrect_experiment;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(12);
  ;;; period_of_superlattice;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(13);
  ;;; period_dispersion;
  [0.0 0.2 0.1 0.3 0.2 0.5 0.4 0.8 0.7 1.0] -> fuzzy_variable(14);

 endif;

enddefmethod;


;;; METHOD 3) build_an_induction_matrix(list_of_consequents);
;;;
;;; The positive and negative induction matrices are built from the data
defmethod build_an_induction_matrix(list_of_consequents);
 lvars list_of_consequents number_of_columns number_of_rows;
 lvars PIT_fuzzy_premises NIT_fuzzy_premises;
 lvars PIT_boolean_premises NIT_boolean_premises;
 lvars PIT_consequents NIT_consequents;
 lvars PIT_fp NIT_fp PIT_bp NIT_bp PIT_c NIT_c;
 lvars counter cn;
 lvars column_number consequent temp1_list temp2_list temp4_list;
 lvars example_fuzzy_premise example_boolean_premise;
 lvars example_consequent temp3_list;
 lvars premise_number value_of_premise membership_function membership;
 lvars check consequent_list value_of_consequent filed;

 pr('   creating induction matrices');nl(1);

 ;;; Set up positive and negative induction matrices for fuzzy premises
 (number_of_fuzzy_premises * 5) + 2 -> number_of_columns;
 (number_of_consequences * 5) -> number_of_rows;
 newarray([1 ^number_of_columns 1 ^number_of_rows])
          -> PIT_fuzzy_premises;
 newarray([1 ^number_of_columns 1 ^number_of_rows])
          -> NIT_fuzzy_premises;
 1 -> column_number;
 until column_number > number_of_columns
 do
  1 -> consequent;
  until consequent > number_of_rows
  do
   0.0 -> PIT_fuzzy_premises(column_number, consequent);
   0.0 -> NIT_fuzzy_premises(column_number, consequent);
   consequent + 1 -> consequent
  enduntil;
  column_number + 1 -> column_number
 enduntil;
```

```
;;; Set up positive and negative induction matrices for Boolean premises
(number_of_boolean_premises * 2) + 2 -> number_of_columns;
newarray([1 ^number_of_columns 1 ^number_of_rows])
                -> PIT_boolean_premises;
newarray([1 ^number_of_columns 1 ^number_of_rows])
                -> NIT_boolean_premises;
1 -> column_number;
until column_number > number_of_columns
do
 1 -> consequent;
 until consequent > number_of_rows
 do
  0.0 -> PIT_boolean_premises(column_number, consequent);
  0.0 -> NIT_boolean_premises(column_number, consequent);
  consequent + 1 -> consequent
 enduntil;
 column_number + 1 -> column_number
enduntil;

;;; Set up temporary induction matrices for fuzzy premises,
;;;  Boolean premises & consequents
(number_of_fuzzy_premises * 5) -> number_of_columns;
newarray([1 ^number_of_columns 1 ^number_of_consequences]) -> PIT_fp;
newarray([1 ^number_of_columns 1 ^number_of_consequences]) -> NIT_fp;

(number_of_boolean_premises * 2) -> number_of_columns;
newarray([1 ^number_of_columns 1 ^number_of_consequences]) -> PIT_bp;
newarray([1 ^number_of_columns 1 ^number_of_consequences]) -> NIT_bp;

newarray([1 5 1 ^number_of_consequences]) -> PIT_c;
newarray([1 5 1 ^number_of_consequences]) -> NIT_c;

;;; Transfer example sets into temporary lists
example_set_fuzzy_premise -> temp1_list;
example_set_boolean_premise -> temp4_list;
example_set_consequent -> temp2_list;

;;; Remove the first example from the example list
until temp1_list = []
do
 hd(temp1_list) -> example_fuzzy_premise;
 hd(temp4_list) -> example_boolean_premise;
 hd(temp2_list) -> example_consequent;

 ;;; Initialize temporary induction matrices for fuzzy premises
 (number_of_fuzzy_premises * 5) -> number_of_columns;
 1 -> column_number;
 until column_number > number_of_columns
 do
  1 -> consequent;
  until consequent > number_of_consequences
  do
   0.0 -> PIT_fp(column_number, consequent);
   0.0 -> NIT_fp(column_number, consequent);
   consequent + 1 -> consequent;
  enduntil;
  column_number + 1 -> column_number
 enduntil;

 ;;; Initialize temporary induction matrices for Boolean premises
 (number_of_boolean_premises * 2) -> number_of_columns;
 1 -> column_number;
 until column_number > number_of_columns
 do
  1 -> consequent;
  until consequent > number_of_consequences
  do
   0.0 -> PIT_bp(column_number, consequent);
   0.0 -> NIT_bp(column_number, consequent);
   consequent + 1 -> consequent;
  enduntil;
  column_number + 1 -> column_number
 enduntil;

 ;;; Initialize temporary induction matrices for consequents
 1 -> column_number;
 until column_number > 5
 do
  1 -> consequent;
  until consequent > number_of_consequences
  do
   0.0 -> PIT_c(column_number, consequent);
   0.0 -> NIT_c(column_number, consequent);
   consequent + 1 -> consequent;
  enduntil;
  column_number + 1 -> column_number
 enduntil;

 ;;; IF a consequent truth value is greater than 0.1
 ;;; THEN store the number of this consequent in the list consequent_list
 [] -> consequent_list;
 1 -> consequent;
 until consequent > number_of_consequences
 do
  if example_consequent(consequent) > 0.1
  then
   consequent_list <> [^consequent] -> consequent_list
  endif;
  consequent + 1 -> consequent
 enduntil;

 ;;; Transfer the list consequent_list into temp3_list
 consequent_list -> temp3_list;
 until temp3_list = []
 do
  hd(temp3_list) -> consequent;
  ;;; if the consequent has been fired enough times in the example set
  ;;; and the value of the consequent in this example is greater than 0.1
  ;;; then ...
  if member(consequent, list_of_consequents)
  then
   ;;; Calculate values of the premises and add to the positive induction
   ;;;  test matrix
   1 -> premise_number;
   until premise_number > number_of_fuzzy_premises
   do
    example_fuzzy_premise(premise_number) -> value_of_premise;
    membership_functions(premise_number) -> membership_function;
    ^calculate_values(value_of_premise, membership_function)
                -> membership;
    (premise_number * 5) - 4 -> column_number;
    membership(1) + PIT_fp(column_number, consequent)
                -> PIT_fp(column_number, consequent);
    column_number + 1 -> column_number;
    membership(2) + PIT_fp(column_number, consequent)
                -> PIT_fp(column_number, consequent);
    column_number + 1 -> column_number;
    membership(3) + PIT_fp(column_number, consequent)
                -> PIT_fp(column_number, consequent);
    column_number + 1 -> column_number;
    membership(4) + PIT_fp(column_number, consequent)
                -> PIT_fp(column_number, consequent);
    column_number + 1 -> column_number;
    membership(5) + PIT_fp(column_number, consequent)
                -> PIT_fp(column_number, consequent);
    premise_number + 1 -> premise_number
   enduntil;

   1 -> premise_number;
   until premise_number > number_of_boolean_premises
   do
    example_boolean_premise(premise_number) -> value_of_premise;
    (premise_number * 2) - 1 -> column_number;
    if value_of_premise = 1
    then
     PIT_bp(column_number, consequent) + 1
                -> PIT_bp(column_number, consequent);
    else
     column_number + 1 -> column_number;
     PIT_bp(column_number, consequent) + 1
                -> PIT_bp(column_number, consequent);
    endif;
    premise_number + 1 -> premise_number;
   enduntil;

   ;;; Calculate values of consequent and add to the positive induction
   ;;;  test matrix
   example_consequent(consequent) -> value_of_consequent;
   fuzzy_variable(consequent) -> membership_function;
   ^calculate_values(value_of_consequent, membership_function)
                -> membership;
   1 -> column_number;
   membership(1) + PIT_c(column_number, consequent)
                -> PIT_c(column_number, consequent);
   column_number + 1 -> column_number;
   membership(2) + PIT_c(column_number, consequent)
                -> PIT_c(column_number, consequent);
   column_number + 1 -> column_number;
```

```
    membership(3) + PIT_c(column_number, consequent)
          -> PIT_c(column_number, consequent);
    column_number + 1 -> column_number;
    membership(4) + PIT_c(column_number, consequent)
          -> PIT_c(column_number, consequent);
    column_number + 1 -> column_number;
    membership(5) + PIT_c(column_number, consequent)
          -> PIT_c(column_number, consequent);
  endif;
  tl(temp3_list) -> temp3_list
enduntil;

1 -> consequent;
until consequent > number_of_consequences
do
  if not(member(consequent, consequent_list))
  then
    if member(consequent, list_of_consequents)
    then
      ;;; Calculate values of the premises and add to the negative induction
      ;;; test matrix
      1 -> premise_number;
      until premise_number > number_of_fuzzy_premises
      do
        example_fuzzy_premise(premise_number) -> value_of_premise;
        membership_functions(premise_number) -> membership_function;
        ;;; METHOD 4:
        "calculate_values(value_of_premise, membership_function)
              -> membership;
        (premise_number * 5) - 4 -> column_number;
        membership(1) + NIT_fp(column_number, consequent)
              -> NIT_fp(column_number, consequent);
        column_number + 1 -> column_number;
        membership(2) + NIT_fp(column_number, consequent)
              -> NIT_fp(column_number, consequent);
        column_number + 1 -> column_number;
        membership(3) + NIT_fp(column_number, consequent)
              -> NIT_fp(column_number, consequent);
        column_number + 1 -> column_number;
        membership(4) + NIT_fp(column_number, consequent)
              -> NIT_fp(column_number, consequent);
        column_number + 1 -> column_number;
        membership(5) + NIT_fp(column_number, consequent)
              -> NIT_fp(column_number, consequent);
        premise_number + 1 -> premise_number
      enduntil;

      1 -> premise_number;
      until premise_number > number_of_boolean_premises
      do
        example_boolean_premise(premise_number) -> value_of_premise;
        (premise_number * 2) - 1 -> column_number;
        if value_of_premise = 1
        then
          NIT_bp(column_number, consequent) + 1
                -> NIT_bp(column_number, consequent);
        else
          column_number + 1 -> column_number;
          NIT_bp(column_number, consequent) + 1
                -> NIT_bp(column_number, consequent);
        endif;
        premise_number + 1 -> premise_number;
      enduntil;

      ;;; Calculate values of consequent and add to the negative induction
      ;;; test matrix
      example_consequent(consequent) -> value_of_consequent;
      fuzzy_variable(consequent) -> membership_function;
      ;;; METHOD 4:
      "calculate_values(value_of_consequent, membership_function)
              -> membership;
      1 -> column_number;
      membership(1) + NIT_c(column_number, consequent)
              -> NIT_c(column_number, consequent);
      column_number + 1 -> column_number;
      membership(2) + NIT_c(column_number, consequent)
              -> NIT_c(column_number, consequent);
      column_number + 1 -> column_number;
      membership(3) + NIT_c(column_number, consequent)
              -> NIT_c(column_number, consequent);
      column_number + 1 -> column_number;
      membership(4) + NIT_c(column_number, consequent)
              -> NIT_c(column_number, consequent);
      column_number + 1 -> column_number;
      membership(5) + NIT_c(column_number, consequent)
              -> NIT_c(column_number, consequent);
    endif;
  endif;
  consequent + 1 -> consequent;
enduntil;

;;; Build up the Positive and Negative Induction Matrices
1 -> consequent;
until consequent > number_of_consequences
do
  1 -> column_number;
  until column_number > 5
  do
    if PIT_c(column_number, consequent) > 0.1
    then
      ((consequent - 1) * 5) + column_number -> cn;
      1 -> counter;
      until counter > (number_of_fuzzy_premises * 5)
      do
        PIT_fuzzy_premises(counter,cn) + PIT_fp(counter,consequent)
              -> PIT_fuzzy_premises(counter,cn);
        counter + 1 -> counter;
      enduntil;

      PIT_fuzzy_premises(counter,cn)+PIT_c(column_number,consequent)
              -> PIT_fuzzy_premises(counter,cn);
      counter + 1 -> counter;
      PIT_fuzzy_premises(counter,cn) + 1
              -> PIT_fuzzy_premises(counter,cn);

      1 -> counter;
      until counter > (number_of_boolean_premises * 2)
      do
        PIT_boolean_premises(counter,cn) + PIT_bp(counter,consequent)
              -> PIT_boolean_premises(counter,cn);
        counter + 1 -> counter
      enduntil;
      PIT_boolean_premises(counter,cn) +
       PIT_c(column_number,consequent)
              -> PIT_boolean_premises(counter,cn);
      counter + 1 -> counter;
      PIT_boolean_premises(counter,cn) + 1
              -> PIT_boolean_premises(counter,cn)

    elseif PIT_c(column_number, consequent) < 0.101
    then
      ((consequent - 1) * 5) + column_number -> cn;

      1 -> counter;
      until counter > (number_of_fuzzy_premises * 5)
      do
        NIT_fuzzy_premises(counter,cn) + NIT_fp(counter,consequent)
              -> NIT_fuzzy_premises(counter,cn);
        counter + 1 -> counter;
      enduntil;

      NIT_fuzzy_premises(counter,cn)+NIT_c(column_number,consequent)
              -> NIT_fuzzy_premises(counter,cn);
      counter + 1 -> counter;
      NIT_fuzzy_premises(counter,cn) + 1
              -> NIT_fuzzy_premises(counter,cn);

      1 -> counter;
      until counter > (number_of_boolean_premises * 2)
      do
        NIT_boolean_premises(counter,cn) + NIT_bp(counter,consequent)
              -> PIT_boolean_premises(counter,cn);
        counter + 1 -> counter
      enduntil;

      NIT_boolean_premises(counter,cn) +
       NIT_c(column_number,consequent)
              -> NIT_boolean_premises(counter,cn);
      counter + 1 -> counter;
      NIT_boolean_premises(counter,cn) + 1
              -> NIT_boolean_premises(counter,cn);
    else
    endif;
    column_number + 1 -> column_number;
  enduntil;
  consequent + 1 -> consequent
enduntil;
```

```
    tl(temp1_list) -> temp1_list;
    tl(temp4_list) -> temp4_list;
    tl(temp2_list) -> temp2_list;
  enduntil;

  pr('    storing induction matrices to file');nl(1);
  if type_of_structure = 'substrate only'
  then
    syscreate("pif1",1,"line") -> filed;
    PIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("pib1",1,"line") -> filed;
    PIT_boolean_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nif1",1,"line") -> filed;
    NIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nib1",1,"line") -> filed;
    NIT_boolean_premises -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'single layer'
  then
    syscreate("pif2",1,"line") -> filed;
    PIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("pib2",1,"line") -> filed;
    PIT_boolean_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nif2",1,"line") -> filed;
    NIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nib2",1,"line") -> filed;
    NIT_boolean_premises -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'multiple layers'
  then
    syscreate("pif3",1,"line") -> filed;
    PIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("pib3",1,"line") -> filed;
    PIT_boolean_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nif3",1,"line") -> filed;
    NIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nib3",1,"line") -> filed;
    NIT_boolean_premises -> datafile(filed);
    sysclose(filed);
  elseif type_of_structure = 'MQW structure'
  then
    syscreate("pif4",1,"line") -> filed;
    PIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("pib4",1,"line") -> filed;
    PIT_boolean_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nif4",1,"line") -> filed;
    NIT_fuzzy_premises -> datafile(filed);
    sysclose(filed);
    syscreate("nib4",1,"line") -> filed;
    NIT_boolean_premises -> datafile(filed);
    sysclose(filed);
  else
  endif;
enddefmethod;


;;; METHOD 4)  calculate_values(value, A);
;;;
;;; Calculates membership values for input value using
;;;  triangular membership functions
defmethod calculate_values(value, A);
  lvars value A size membership;
  lvars count i start finish halfway midpoint;

  round(length(A) / 2.0) -> size;
  newarray([1 ^size]) -> membership;

  1 -> count;
  1 -> i;
  until i>10
  do
    A(i) -> start;
    i+1 -> i;
    A(i) -> finish;

    (finish - start)/2.0 -> halfway;
    start + halfway -> midpoint;

    if value < midpoint
    then
      (value - start) / (midpoint - start) -> membership(count)
    else
      ((midpoint - value) / (finish - midpoint)) + 1.0 -> membership(count)
    endif;

    if membership(count) > 1.0 or membership(count) < 0.0
    then
      0.0 -> membership(count)
    endif;

    count+1 -> count;
    i+1 -> i
  enduntil;

  return(membership);
enddefmethod;


;;; METHOD 5)  get_lists_of_potential_rules(list_of_consequents);
;;;
;;; Using positive and negative thresholds, potential rules are induced
;;;  from the induction matrices. These rules are stored in an array
;;;  called potential_rules.
defmethod get_lists_of_potential_rules(list_of_consequents);
  lvars list_of_consequents filed i j;
  lvars PIT_fuzzy_premises NIT_fuzzy_premises PIT_boolean_premises;
  lvars NIT_boolean_premises PIT_consequents NIT_consequents;
  lvars all_the_fuzzy_premises all_the_boolean_premises;
  lvars consequent column_number count;
  lvars positive_threshold negative_threshold temp_list_P;
  lvars temp_list_C_1 temp_list_C_2;
  lvars value last_consequent temp_list_P_2;
  lvars number_of_consequents count2 cn ct vl;

  if type_of_structure = 'substrate only'
  then
    sysopen("pif1",0,"line") -> filed;
    datafile(filed) -> PIT_fuzzy_premises;
    sysclose(filed);
    sysopen("nif1",0,"line") -> filed;
    datafile(filed) -> NIT_fuzzy_premises;
    sysclose(filed);
    sysopen("pib1",0,"line") -> filed;
    datafile(filed) -> PIT_boolean_premises;
    sysclose(filed);
    sysopen("nib1",0,"line") -> filed;
    datafile(filed) -> NIT_boolean_premises;
    sysclose(filed);
  elseif type_of_structure = 'single layer'
  then
    sysopen("pif2",0,"line") -> filed;
    datafile(filed) -> PIT_fuzzy_premises;
    sysclose(filed);
    sysopen("nif2",0,"line") -> filed;
    datafile(filed) -> NIT_fuzzy_premises;
    sysclose(filed);
    sysopen("pib2",0,"line") -> filed;
    datafile(filed) -> PIT_boolean_premises;
    sysclose(filed);
    sysopen("nib2",0,"line") -> filed;
    datafile(filed) -> NIT_boolean_premises;
    sysclose(filed);
  elseif type_of_structure = 'multiple layers'
  then
    sysopen("pif3",0,"line") -> filed;
    datafile(filed) -> PIT_fuzzy_premises;
    sysclose(filed);
    sysopen("nif3",0,"line") -> filed;
    datafile(filed) -> NIT_fuzzy_premises;
    sysclose(filed);
    sysopen("pib3",0,"line") -> filed;
    datafile(filed) -> PIT_boolean_premises;
    sysclose(filed);
    sysopen("nib3",0,"line") -> filed;
    datafile(filed) -> NIT_boolean_premises;
    sysclose(filed);
  elseif type_of_structure = 'MQW structure'
```

```
then
  sysopen("pif4",0,"line") -> filed;
  datafile(filed) -> PIT_fuzzy_premises;
  sysclose(filed);
  sysopen("nif4",0,"line") -> filed;
  datafile(filed) -> NIT_fuzzy_premises;
  sysclose(filed);
  sysopen("pib4",0,"line") -> filed;
  datafile(filed) -> PIT_boolean_premises;
  sysclose(filed);
  sysopen("nib4",0,"line") -> filed;
  datafile(filed) -> NIT_boolean_premises;
  sysclose(filed);
else
endif;

number_of_consequences * 5 -> number_of_consequents;
newarray([1 ^number_of_consequents 1 4]) -> potential_rules;
1 -> i;
until i > number_of_consequents
do
  for j in [1 2 3 4]
  do
    [] -> potential_rules(i,j);
  endfor;
  i + 1 -> i
enduntil;

number_of_fuzzy_premises * 5 -> all_the_fuzzy_premises;
number_of_boolean_premises * 2 -> all_the_boolean_premises;

1 -> count;
1 -> count2;
1 -> consequent;
until consequent > number_of_consequents
do
  1 -> ct;
  consequent -> vl;
  until vl < 6
  do
    vl - 6 -> vl;
    ct + 1 -> ct
  enduntil;

  all_the_fuzzy_premises + 1 -> cn;
  if PIT_fuzzy_premises(cn, consequent) > 0.0
        and PIT_fuzzy_premises(cn+1, consequent) > 5
  then
    if effectiveness(ct) = 0
    then
      thresholds * PIT_fuzzy_premises(cn+1, consequent)
          -> positive_threshold;
      neg * PIT_fuzzy_premises(cn+1, consequent) -> negative_threshold;
    else
      (thresholds - 0.2) * PIT_fuzzy_premises(cn+1, consequent)
          -> positive_threshold;
      (neg + 0.1) * PIT_fuzzy_premises(cn+1, consequent)
          -> negative_threshold;
    endif;

    [] -> temp_list_P;
    [] -> temp_list_P_2;
    [] -> temp_list_C_1;
    [] -> temp_list_C_2;

    1 -> column_number;
    until column_number > all_the_fuzzy_premises
    do
      if PIT_fuzzy_premises(column_number, consequent)
          > positive_threshold and
        NIT_fuzzy_premises(column_number, consequent)
          < negative_threshold
      then
        temp_list_P <> [^column_number] -> temp_list_P
      endif;
      column_number + 1 -> column_number
    enduntil;

    PIT_fuzzy_premises(cn+1, consequent) - 1 -> positive_threshold;
    1 -> negative_threshold;

    1 -> column_number;
    until column_number > all_the_boolean_premises
    do
      if PIT_boolean_premises(column_number, consequent)
          > positive_threshold and
        NIT_boolean_premises(column_number, consequent)
          < negative_threshold
      then
        temp_list_P_2 <> [^column_number] -> temp_list_P_2
      endif;
      column_number + 1 -> column_number;

    enduntil;

    if temp_list_P = [] and temp_list_P_2 = []
    then
          [] -> temp_list_C_1;
          [] -> temp_list_C_2;
    else
      [^count] -> temp_list_C_1;
      [^count2] -> temp_list_C_2;
      potential_rules(consequent,1) <> [^^temp_list_P]
            -> potential_rules(consequent,1);
      potential_rules(consequent,2) <> [^^temp_list_P_2]
            -> potential_rules(consequent,2);
      potential_rules(consequent,3) <> [^^temp_list_C_1]
            -> potential_rules(consequent,3);
      potential_rules(consequent,4) <> [^^temp_list_C_2]
            -> potential_rules(consequent,4);

      if not(potential_rules(consequent,1) = []
          and potential_rules(consequent,2) = [])
      then
        number_of_potential_rules + 1 -> number_of_potential_rules
      endif;
    endif;
  endif;
  consequent + 1 -> consequent;
  count + 1 -> count;
  if count > 5
  then
    1 -> count;
    count2 + 1 -> count2
  endif;
enduntil;

1 -> consequent;
until consequent > number_of_consequents
do
  if length(potential_rules(consequent,1)) > 2
  then
    [] -> potential_rules(consequent,1);
    [] -> potential_rules(consequent,2);
    [] -> potential_rules(consequent,3);
    [] -> potential_rules(consequent,4);
  endif;
  consequent+1 -> consequent
enduntil;

if type_of_structure = 'substrate only'
then
  syscreate("potrule1",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif type_of_structure = 'single layer'
then
  syscreate("potrule2",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
  syscreate("some1",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif type_of_structure = 'multiple layers'
then
  syscreate("potrule3",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif type_of_structure = 'MQW structure'
then
  syscreate("potrule4",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
else
endif;
enddefmethod;
```

```
;;; METHOD 6) check_if_potential_rules_already_exists(i);
;;;
;;; Checks through existing rules and sees if potential rules
;;;  already exist
defmethod check_if_potential_rules_already_exists;
 lvars filed i number_of_rules;
 lvars existing_rules potential_rules rule_number consequent;

  if i = 1
  then
    sysopen("rules1",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    sysopen("rno1",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("potrule1",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
  elseif i = 2
  then
    sysopen("rules2",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    sysopen("rno2",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("potrule2",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
  elseif i = 3
  then
    sysopen("rules3",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    sysopen("rno3",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("potrule3",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
  else
    sysopen("rules4",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    sysopen("rno4",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("potrule4",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
  endif;

  1 -> rule_number;
  until rule_number > number_of_rules
  do
    existing_rules(rule_number,4) -> consequent;
    if consequent = []
    then
    else
      hd(consequent) -> consequent;

      if existing_rules(rule_number,1) = potential_rules(consequent,1) and
        existing_rules(rule_number,2) = potential_rules(consequent,2) and
        existing_rules(rule_number,3) = potential_rules(consequent,3) and
        existing_rules(rule_number,4) = potential_rules(consequent,4)
      then
        [] -> potential_rules(consequent,1);
        [] -> potential_rules(consequent,2);
        [] -> potential_rules(consequent,3);
        [] -> potential_rules(consequent,4)
      endif;

      if subset_of(existing_rules(rule_number,1),
            potential_rules(consequent,1)) or
        subset_of(existing_rules(rule_number,2),
            potential_rules(consequent,2))
      then
        [] -> potential_rules(consequent,1);
        [] -> potential_rules(consequent,2);
        [] -> potential_rules(consequent,3);
        [] -> potential_rules(consequent,4)
      endif;
    endif;
    rule_number + 1 -> rule_number
  enduntil;

  if i = 1
  then
    syscreate("potrule1",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif i = 2
  then
    syscreate("potrule2",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
    syscreate("some2",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif i = 3
  then
    syscreate("potrule3",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif i = 4
  then
    syscreate("potrule4",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  else
  endif;
enddefmethod;


;;; METHOD 7) check_for_coverage(quota, high_cardinality);
;;;
;;; This method checks all potential rules for coverage
defmethod check_for_coverage(quota, high_cardinality);
 lvars quota high_cardinality consequent cardinality;
 lvars filed i consequents_in_rules potential_rules;
 lvars PIT_fuzzy_premises PIT_boolean_premises;
 lvars temp1_list temp2_list temp_list;
 lvars check value count cn cum_con;

  if type_of_structure='substrate only'
  then
    sysopen("cinr1",0,"line") -> filed;
    datafile(filed) -> consequents_in_rules;
    sysclose(filed);
    4 -> number_of_fuzzy_premises;
    7 -> number_of_consequences;
    sysopen("potrule1",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
    sysopen("pif1",0,"line") -> filed;
    datafile(filed) -> PIT_fuzzy_premises;
    sysclose(filed);
    sysopen("pib1",0,"line") -> filed;
    datafile(filed) -> PIT_boolean_premises;
    sysclose(filed);
  elseif type_of_structure='single layer'
  then
    sysopen("cinr2",0,"line") -> filed;
    datafile(filed) -> consequents_in_rules;
    sysclose(filed);
    7 -> number_of_fuzzy_premises;
    18 -> number_of_consequences;
    sysopen("potrule2",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
    sysopen("pif2",0,"line") -> filed;
    datafile(filed) -> PIT_fuzzy_premises;
    sysclose(filed);
    sysopen("pib2",0,"line") -> filed;
    datafile(filed) -> PIT_boolean_premises;
    sysclose(filed);
  elseif type_of_structure='multiple layers'
  then
    sysopen("cinr3",0,"line") -> filed;
    datafile(filed) -> consequents_in_rules;
    sysclose(filed);
    6 -> number_of_fuzzy_premises;
    10 -> number_of_consequences;
    sysopen("potrule3",0,"line") -> filed;
    datafile(filed) -> potential_rules;
    sysclose(filed);
```

```
  sysopen("pif3",0,"line") -> filed;
  datafile(filed) -> PIT_fuzzy_premises;
  sysclose(filed);
  sysopen("pib3",0,"line") -> filed;
  datafile(filed) -> PIT_boolean_premises;
  sysclose(filed);
 elseif type_of_structure='MQW structure'
 then
  sysopen("cinr4",0,"line") -> filed;
  datafile(filed) -> consequents_in_rules;
  sysclose(filed);
  8 -> number_of_fuzzy_premises;
  14 -> number_of_consequences;
  sysopen("potrule4",0,"line") -> filed;
  datafile(filed) -> potential_rules;
  sysclose(filed);
  sysopen("pif4",0,"line") -> filed;
  datafile(filed) -> PIT_fuzzy_premises;
  sysclose(filed);
  sysopen("pib4",0,"line") -> filed;
  datafile(filed) -> PIT_boolean_premises;
  sysclose(filed);
 else
 endif;

 (number_of_fuzzy_premises * 5) + 1 -> cum_con;

 1 -> consequent;
 until consequent > number_of_consequences * 5
 do
  potential_rules(consequent,1) -> temp1_list;
  potential_rules(consequent,2) -> temp2_list;

  1 -> count;
  consequent -> cn;
  until cn < 6
  do
   cn - 6 -> cn;
   count + 1 -> count
  enduntil;

  if not(temp1_list = [] and temp2_list = [])
     and consequents_in_rules(count) > quota
  then
   hd(potential_rules(consequent,3)) -> value;
   PIT_fuzzy_premises(cum_con,consequent) * high_cardinality / 100
            -> cardinality;

   0 -> check;
   until temp1_list = []
   do
    hd(temp1_list) -> value;
    if PIT_fuzzy_premises(value,consequent) < cardinality
    then
     1 -> check;
    endif;
    tl(temp1_list) -> temp1_list;
   enduntil;
   until temp2_list = []
   do
    hd(temp2_list) -> value;
    if PIT_boolean_premises(value,consequent) < cardinality
    then
     1 -> check;
    endif;
    tl(temp2_list) -> temp2_list;
   enduntil;

   if check = 1
   then
    [] -> potential_rules(consequent,1);
    [] -> potential_rules(consequent,2);
    [] -> potential_rules(consequent,3);
    [] -> potential_rules(consequent,4)
   endif;
  endif;
  consequent + 1 -> consequent;
 enduntil;

 if i = 1
 then
  syscreate("potrule1",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
 elseif i = 2
 then
  syscreate("potrule2",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
  syscreate("some3",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
 elseif i = 3
 then
  syscreate("potrule3",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
 elseif i = 4
 then
  syscreate("potrule4",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
 else
 endif;
enddefmethod;


;;; METHOD 8) display_new_rules(i);
;;;
;;; This method displays new rules for user to decide whether
;;; to accept them

defmethod display_new_rules(i);
 lvars i filed number_of_consequences potential_rules;
 lvars consequent temp_list temp_list_2 value number_of_variable;
 lvars truth_value linguistic_value check keep_it;

  if i = 1
  then
   7 -> number_of_consequences;
   sysopen("potrule1",0,"line") -> filed;
   datafile(filed) -> potential_rules;
   sysclose(filed);
  elseif i = 2
  then
   18 -> number_of_consequences;
   sysopen("potrule2",0,"line") -> filed;
   datafile(filed) -> potential_rules;
   sysclose(filed);
  elseif i = 3
  then
   7 -> number_of_consequences;
   sysopen("potrule3",0,"line") -> filed;
   datafile(filed) -> potential_rules;
   sysclose(filed);
  elseif i = 4
  then
   14 -> number_of_consequences;
   sysopen("potrule4",0,"line") -> filed;
   datafile(filed) -> potential_rules;
   sysclose(filed);
  else
  endif;

  1 -> consequent;
  until consequent > number_of_consequences * 5
  do
   0 -> check;
   if not(potential_rules(consequent,1) = [])
   then
    nl(2);
    pr('_____');nl(2);
    pr('FUZZY RULE:');
    potential_rules(consequent,2) -> temp_list;
    nl(2);pr('IF  ');nl(1);
    until temp_list = []
    do
     hd(temp_list) -> value;

     1 -> number_of_variable;
     value -> truth_value;
     until truth_value < 3
     do
      truth_value - 2 -> truth_value;
      number_of_variable + 1 -> number_of_variable
     enduntil;

     if truth_value = 1
```

```
  then
    if check = 1
    then
      nl(1);pr(' AND');nl(1);
    endif;
    1 -> check;

    pr(boolean_premises(number_of_variable));pr(' = ');

    if truth_value = 1
    then
      pr(' TRUE ')
    else
      pr(' FALSE ')
    endif;

    1 -> check;
  endif;
  tl(temp_list) -> temp_list
enduntil;

potential_rules(consequent,1) -> temp_list;
until temp_list = []
do
  if check = 1
  then
    nl(1);pr(' AND');nl(1);
  endif;
  1 -> check;

  hd(temp_list) -> value;

  1 -> number_of_variable;
  value -> linguistic_value;
  until linguistic_value < 6
  do
    linguistic_value - 5 -> linguistic_value;
    number_of_variable + 1 -> number_of_variable
  enduntil;

  pr(fuzzy_premises(number_of_variable));pr(' = ');

  if linguistic_value = 1
  then
    pr(' NONE ')
  elseif linguistic_value = 2
  then
    pr(' SOME ')
  elseif linguistic_value = 3
  then
    pr(' FAIRLY ')
  elseif linguistic_value = 4
  then
    pr(' VERY ')
  elseif linguistic_value = 5
  then
    pr(' EXTREME ')
  else
  endif;
  tl(temp_list) -> temp_list;
enduntil;

potential_rules(consequent,3) -> temp_list;
potential_rules(consequent,4) -> temp_list_2;
nl(2);pr('THEN   ');
until temp_list = []
do
  hd(temp_list_2) -> value;
  pr(consequences(value));pr('  =  ');
  hd(temp_list) -> value;
  if value = 1
  then
    pr(' NONE ')
  elseif value = 2
  then
    pr(' SOME ')
  elseif value = 3
  then
    pr(' FAIRLY ')
  elseif value = 4
  then
    pr(' VERY ')
  elseif value = 5
  then
    pr(' EXTREME ')
  else
  endif;
  tl(temp_list) -> temp_list;
enduntil;
nl(3);pr(' Do you wish to add this rule to the fuzzy ruleset (Y/N) ');
readline() -> keep_it;
if not(keep_it = [Y] or keep_it = [y])
then
  [] -> potential_rules(consequent,1);
  [] -> potential_rules(consequent,2);
  [] -> potential_rules(consequent,3);
  [] -> potential_rules(consequent,4)
endif;
endif;
consequent + 1 -> consequent;
enduntil;

if i = 1
then
  syscreate("potrule1",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif i = 2
then
  syscreate("potrule2",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
  syscreate("some3",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif i = 3
then
  syscreate("potrule3",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
elseif i = 4
then
  syscreate("potrule4",1,"line") -> filed;
  potential_rules -> datafile(filed);
  sysclose(filed);
else
endif;

nl(3);
enddefmethod;


;;;  METHOD 9)  add_rules_to_rulebase;
;;;
;;;  Add new rules to the fuzzy rulebase
;;;  and change control variables appropriately

defmethod add_rules_to_rulebase;
  lvars s i e filed new_rules existing_rules;
  lvars number_of_consequences number_of_consequents;
  lvars new_ruleset number_of_rules new_number_of_rules;
  lvars consequent list_of_rules value value2 count count2;
  lvars number_of_experts connection_matrix rules_for_substrate_only;
  lvars rules_for_single_layer rules_for_multiple_layers rules_for_MQW;
  lvars list_of_fuzzy_variables number_of_fuzzy_variables;
  lvars new_number_of_fuzzy_variables;
  lvars temp_list rule_history r_hist masking_of_rule mask_rule;
  lvars fuzzy_rules mf mf2 layers variable_history var_hist var_changes var_ch;

  nl(2);pr('_____');
  nl(3);pr('Add new fuzzy rules to the rulebase.');nl(1);

  sysopen("noexp",0,"line") -> filed;
  datafile(filed) -> number_of_experts;
  sysclose(filed);

  make_instance([CONNECTION_MATRIX]) -> connection_matrix;

  for s in [1 2 3 4]
  do
    if s = 1
    then
      sysopen("noffc1",0,"line") -> filed;
      datafile(filed) -> number_of_consequences;
      sysclose(filed);
      sysopen("potrule1",0,"line") -> filed;
      datafile(filed) -> new_rules;
      sysclose(filed);
```

```
  sysopen("rno1",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
  sysopen("rules1",0,"line") -> filed;
  datafile(filed) -> existing_rules;
  sysclose(filed);
  if e = 1
  then
    sysopen("lofr1s1",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 2
  then
    sysopen("lofr2s1",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 3
  then
    sysopen("lofr3s1",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 4
  then
    sysopen("lofr4s1",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  else
    sysopen("lofr5s1",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  endif;
elseif s = 2
then
  sysopen("noffc2",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("potrule2",0,"line") -> filed;
  datafile(filed) -> new_rules;
  sysclose(filed);
  sysopen("rno2",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
  sysopen("rules2",0,"line") -> filed;
  datafile(filed) -> existing_rules;
  sysclose(filed);
  if e = 1
  then
    sysopen("lofr1s2",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 2
  then
    sysopen("lofr2s2",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 3
  then
    sysopen("lofr3s2",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 4
  then
    sysopen("lofr4s2",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  else
    sysopen("lofr5s2",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  endif;
elseif s = 3
then
  sysopen("noffc3",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("potrule3",0,"line") -> filed;
  datafile(filed) -> new_rules;
  sysclose(filed);
  sysopen("rno3",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
  sysopen("rules3",0,"line") -> filed;
  datafile(filed) -> existing_rules;
  sysclose(filed);
  if e = 1
  then
    sysopen("lofr1s3",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 2
  then
    sysopen("lofr2s3",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 3
  then
    sysopen("lofr3s3",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 4
  then
    sysopen("lofr4s3",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  else
    sysopen("lofr5s3",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  endif;
elseif s = 4
then
  sysopen("noffc4",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("potrule4",0,"line") -> filed;
  datafile(filed) -> new_rules;
  sysclose(filed);
  sysopen("rno4",0,"line") -> filed;
  datafile(filed) -> number_of_rules;
  sysclose(filed);
  sysopen("rules4",0,"line") -> filed;
  datafile(filed) -> existing_rules;
  sysclose(filed);
  if e = 1
  then
    sysopen("lofr1s4",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 2
  then
    sysopen("lofr2s4",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 3
  then
    sysopen("lofr3s4",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  elseif e = 4
  then
    sysopen("lofr4s4",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  else
    sysopen("lofr5s4",0,"line") -> filed;
    datafile(filed) -> list_of_rules;
    sysclose(filed);
  endif;
else
endif;

number_of_consequences * 5 -> number_of_consequents;

;;; Calculate the total number of rules when new rules are added
number_of_rules -> new_number_of_rules;
1 -> consequent;
until consequent > no_of_rules
do
  if not(new_rules(consequent,1) = [] and new_rules(consequent,2) = [])
  then
    new_number_of_rules + 1 -> new_number_of_rules
  endif;
  consequent + 1 -> consequent
enduntil;

;;; Add the existing rules to the new ruleset
newarray([1 ^new_number_of_rules 1 4]) -> new_ruleset;
1 -> i;
```

```
until i > number_of_rules
do
  existing_rules(i,1) -> new_ruleset(i,1);
  existing_rules(i,2) -> new_ruleset(i,2);
  existing_rules(i,3) -> new_ruleset(i,3);
  existing_rules(i,4) -> new_ruleset(i,4);
  i+1 -> i
enduntil;

[] -> temp_list;
[] -> list_of_fuzzy_variables;
;;; Add new rules into the new ruleset
1 -> consequent;
until consequent > no_of_rules
do
  if not(new_rules(consequent,1) = [] and new_rules(consequent,2) = [])
  then
    new_rules(consequent,1) -> new_ruleset(i,1);
    new_rules(consequent,2) -> new_ruleset(i,2);
    new_rules(consequent,3) -> new_ruleset(i,3);
    new_rules(consequent,4) -> new_ruleset(i,4);
    ;;; Make a list the numbers of fuzzy premises in each new rule
    length(new_rules(consequent,1)) -> value;
    temp_list <> [^value] -> temp_list;
    ;;; Record the fuzzy premises in a list
    new_rules(consequent,1) -> value2;
    list_of_fuzzy_variables <> [^^value2] -> list_of_fuzzy_variables;
    list_of_rules <> [^i] -> list_of_rules;
    i+1 -> i
  endif;
  consequent + 1 -> consequent
enduntil;

if s = 1
then
  sysopen("rhist1",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
  sysopen("rmask1",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
  sysopen("fuzzno1",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_variables;
  sysclose(filed);
  sysopen("frules1",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist1",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch1",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
  sysopen("memf1",0,"line") -> filed;
  datafile(filed) -> mf;
  sysclose(filed);
  sysopen("l1",0,"line") -> filed;
  datafile(filed) -> layers;
  sysclose(filed);

  ;;;  Substrate Only
  sysopen("lofr1s1",0,"line") -> filed;
  datafile(filed) -> rules_for_substrate_only;
  sysclose(filed);
  if e = 1
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr1s1",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
          rules_for_substrate_only);
  connection_matrix <- save_to_file(1,1);

  sysopen("lofr2s1",0,"line") -> filed;
  datafile(filed) -> rules_for_substrate_only;
  sysclose(filed);
  if e = 2
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr2s1",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
  connection_matrix <- save_to_file(2,1);

  sysopen("lofr3s1",0,"line") -> filed;
  datafile(filed) -> rules_for_substrate_only;
  sysclose(filed);
  if e = 3
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr3s1",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
  connection_matrix <- save_to_file(3,1);

  sysopen("lofr4s1",0,"line") -> filed;
  datafile(filed) -> rules_for_substrate_only;
  sysclose(filed);
  if e = 4
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr4s1",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
  connection_matrix <- save_to_file(4,1);
  if e = 5
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr5s1",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
  connection_matrix <- save_to_file(5,1);

elseif s = 2
then
  sysopen("rhist2",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
  sysopen("rmask2",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
  sysopen("fuzzno2",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_variables;
  sysclose(filed);
  sysopen("frules2",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist2",0,"line") -> filed;
  datafile(filed) -> variable_history;
```

```
sysclose(filed);
sysopen("varch2",0,"line") -> filed;
datafile(filed) -> var_changes;
sysclose(filed);
sysopen("memf2",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);
sysopen("l2",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);

;;; Single Layer
sysopen("lofr1s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 1
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer < > [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr1s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
    rules_for_single_layer);
connection_matrix <- save_to_file(1,2);

sysopen("lofr2s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 2
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer < > [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr2s2",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
    rules_for_single_layer);
connection_matrix <- save_to_file(2,2);

sysopen("lofr3s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 3
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer < > [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr3s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
    rules_for_single_layer);
connection_matrix <- save_to_file(3,2);

sysopen("lofr4s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 4
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer < > [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr4s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
    rules_for_single_layer);
connection_matrix <- save_to_file(4,2);
if e = 5
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer < > [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr5s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
    rules_for_single_layer);
connection_matrix <- save_to_file(5,2);

elseif s = 3
then
  sysopen("rhist3",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
  sysopen("rmask3",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
  sysopen("fuzzno3",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_variables;
  sysclose(filed);
  sysopen("frules3",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhist3",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch3",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
  sysopen("memf3",0,"line") -> filed;
  datafile(filed) -> mf;
  sysclose(filed);
  sysopen("l3",0,"line") -> filed;
  datafile(filed) -> layers;
  sysclose(filed);

  ;;; Multiple Layers
  sysopen("lofr1s3",0,"line") -> filed;
  datafile(filed) -> rules_for_multiple_layers;
  sysclose(filed);
  if e = 1
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr1s3",1,"line") -> filed;
  rules_for_multiple_layers -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
  connection_matrix <- save_to_file(1,3);

  sysopen("lofr2s3",0,"line") -> filed;
  datafile(filed) -> rules_for_multiple_layers;
  sysclose(filed);
  if e = 2
  then
    number_of_rules + 1 -> count;
    until count > new_number_of_rules
    do
      rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
      count + 1 -> count
    enduntil;
  endif;
  syscreate("lofr2s3",1,"line") -> filed;
  rules_for_substrate_only -> datafile(filed);
  sysclose(filed);
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
  connection_matrix <- save_to_file(2,3);
```

```
sysopen("lofr3s3",0,"line") -> filed;
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 3
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr3s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(3,3);

sysopen("lofr4s3",0,"line") -> filed;
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 4
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr4s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(4,3);
if e = 5
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr5s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(5,3);

elseif s = 4
then
  sysopen("rhist4",0,"line") -> filed;
  datafile(filed) -> rule_history;
  sysclose(filed);
  sysopen("rmask4",0,"line") -> filed;
  datafile(filed) -> masking_of_rule;
  sysclose(filed);
  sysopen("fuzzno4",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_variables;
  sysclose(filed);
  sysopen("frules4",0,"line") -> filed;
  datafile(filed) -> fuzzy_rules;
  sysclose(filed);
  sysopen("varhis4",0,"line") -> filed;
  datafile(filed) -> variable_history;
  sysclose(filed);
  sysopen("varch4",0,"line") -> filed;
  datafile(filed) -> var_changes;
  sysclose(filed);
  sysopen("memf4",0,"line") -> filed;
  datafile(filed) -> mf;
  sysclose(filed);
  sysopen("l4",0,"line") -> filed;
  datafile(filed) -> layers;
  sysclose(filed);

  ;;;  MQW Structure and Superlattices
  sysopen("lofr1s4",0,"line") -> filed;
  datafile(filed) -> rules_for_MQW;
  sysclose(filed);
if e = 1
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_MQW < > [^count] -> rules_for_MQW;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr1s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_MQW);
connection_matrix <- save_to_file(1,4);

sysopen("lofr2s4",0,"line") -> filed;
datafile(filed) -> rules_for_MQW;
sysclose(filed);
if e = 2
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_MQW < > [^count] -> rules_for_MQW;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr2s4",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_MQW);
connection_matrix <- save_to_file(2,4);

sysopen("lofr3s4",0,"line") -> filed;
datafile(filed) -> rules_for_MQW;
sysclose(filed);
if e = 3
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_MQW < > [^count] -> rules_for_MQW;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr3s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_MQW);
connection_matrix <- save_to_file(3,4);

sysopen("lofr4s4",0,"line") -> filed;
datafile(filed) -> rules_for_MQW;
sysclose(filed);
if e = 4
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_MQW < > [^count] -> rules_for_MQW;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr4s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_MQW);
connection_matrix <- save_to_file(4,4);
if e = 5
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_MQW < > [^count] -> rules_for_MQW;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr5s4",1,"line") -> filed;
rules_for_MQW -> datafile(filed);
sysclose(filed);
```

```
  connection_matrix<-create_a_connection_matrix(new_number_of_rules,
       rules_for_MQW);
  connection_matrix <- save_to_file(5,4);
 else
 endif;

 newarray([1 ^new_number_of_rules]) -> r_hist;
 newarray([1 ^new_number_of_rules]) -> mask_rule;
 1 -> i;
 until i > number_of_rules
 do
  rule_history(i) -> r_hist(i);
  masking_of_rule(i) -> mask_rule(i);
  i+1 -> i
 enduntil;
 until i > new_number_of_rules
 do
  [] -> r_hist(i);
  "ASSERT" -> mask_rule(i);
  i+1 -> i
 enduntil;

 number_of_fuzzy_variables -> new_number_of_fuzzy_variables;
 number_of_rules + 1 -> count;
 until count > new_number_of_rules
 do
  hd(temp_list) -> value;
  fuzzy_rules(value,1) <> [^count] -> fuzzy_rules(value,1);
  1 -> count2;
  until count2 > value
  do
   new_number_of_fuzzy_variables + 1
              -> new_number_of_fuzzy_variables;
   fuzzy_rules(value,2) <> [^new_number_of_fuzzy_variables]
              -> fuzzy_rules(value,2);
   count2 + 1 -> count2
  enduntil;
  count + 1 -> count;
  tl(temp_list) -> temp_list
 enduntil;

 newarray([1 2 1 ^new_number_of_fuzzy_variables]) -> var_hist;
 newarray([1 ^new_number_of_fuzzy_variables]) -> var_ch;
 newarray([1 ^new_number_of_fuzzy_variables]) -> mf2;
 1 -> i;
 until i > number_of_fuzzy_variables
 do
  variable_history(1,i) -> var_hist(1,i);
  variable_history(2,i) -> var_hist(2,i);
  var_changes(i) -> var_ch(i);
  mf(i) -> mf2(i);
  i+1 -> i
 enduntil;
 until list_of_fuzzy_variables = []
 do
  [] -> var_hist(1,i);
  [] -> var_hist(2,i);
  [] -> var_ch(i);
  hd(list_of_fuzzy_variables) -> value;
  1 -> count;
  until value < 6
  do
   value - 5 -> value;
   count + 1 -> count;
  enduntil;
  count -> mf2(i);
  layers <> [1] -> layers;
  i+1 ->i;
  tl(list_of_fuzzy_variables) -> list_of_fuzzy_variables
 enduntil;

 if s = 1
 then
  syscreate("rno1",1,"line") -> filed;
  new_number_of_rules -> datafile(filed);
  sysclose(filed);
  syscreate("rules1",1,"line") -> filed;
  new_rules -> datafile(filed);
  sysclose(filed);
  if e=1
  then
   syscreate("lofr1s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=2
  then
   syscreate("lofr2s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=3
  then
   syscreate("lofr3s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=4
  then
   syscreate("lofr4s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=5
  then
   syscreate("lofr5s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  else
  endif;
  syscreate("rhist1",1,"line") -> filed;
  r_hist -> datafile(filed);
  sysclose(filed);
  syscreate("rmask1",1,"line") -> filed;
  mask_rule -> datafile(filed);
  sysclose(filed);
  syscreate("fuzzno1",1,"line") -> filed;
  new_number_of_fuzzy_variables -> datafile(filed);
  sysclose(filed);
  syscreate("frules1",1,"line") -> filed;
  fuzzy_rules -> datafile(filed);
  sysclose(filed);
  syscreate("varhist1",1,"line") -> filed;
  var_hist -> datafile(filed);
  sysclose(filed);
  syscreate("varch1",1,"line") -> filed;
  var_ch -> datafile(filed);
  sysclose(filed);
  syscreate("memf1",1,"line") -> filed;
  mf2 -> datafile(filed);
  sysclose(filed);
  syscreate("l1",1,"line") -> filed;
  layers -> datafile(filed);
  sysclose(filed);
 elseif s = 2
 then
  syscreate("rno2",1,"line") -> filed;
  new_number_of_rules -> datafile(filed);
  sysclose(filed);
  syscreate("rules2",1,"line") -> filed;
  new_rules -> datafile(filed);
  sysclose(filed);
  if e=1
  then
   syscreate("lofr1s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=2
  then
   syscreate("lofr2s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=3
  then
   syscreate("lofr3s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=4
  then
   syscreate("lofr4s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=5
  then
   syscreate("lofr5s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  else
  endif;
  syscreate("rhist2",1,"line") -> filed;
  r_hist -> datafile(filed);
  sysclose(filed);
```

```
    syscreate("rmask2",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno2",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules2",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist2",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch2",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf2",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l2",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  elseif s = 3
  then
    syscreate("rno3",1,"line") -> filed;
    new_number_of_rules -> datafile(filed);
    sysclose(filed);
    syscreate("rules3",1,"line") -> filed;
    new_rules -> datafile(filed);
    sysclose(filed);
    if e=1
    then
      syscreate("lofr1s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=2
    then
      syscreate("lofr2s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=3
    then
      syscreate("lofr3s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=4
    then
      syscreate("lofr4s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=5
    then
      syscreate("lofr5s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    else
    endif;
    syscreate("rhist3",1,"line") -> filed;
    r_hist -> datafile(filed);
    sysclose(filed);
    syscreate("rmask3",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno3",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules3",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist3",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch3",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf3",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l3",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  elseif s = 4
  then
    syscreate("rno4",1,"line") -> filed;
    new_number_of_rules -> datafile(filed);
    sysclose(filed);
    syscreate("rules4",1,"line") -> filed;
    new_rules -> datafile(filed);
    sysclose(filed);
    if e=1
    then
      syscreate("lofr1s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=2
    then
      syscreate("lofr2s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=3
    then
      syscreate("lofr3s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=4
    then
      syscreate("lofr4s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=5
    then
      syscreate("lofr5s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    else
    endif;
    syscreate("rhist4",1,"line") -> filed;
    r_hist -> datafile(filed);
    sysclose(filed);
    syscreate("rmask4",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno4",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules4",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist4",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch4",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf4",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l4",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  else
  endif;
endfor;
enddefmethod;

endflavour;
```

[12] The program 'editrule.p' is a simple rule editor for adding new rules to the fuzzy rulebase.

```
;;; PROGRAM NAME: editrule.p

;;; Simple rule editor for adding rules to the fuzzy system

;;; INDEX OF OBJECTS
;;;
;;; 1. Function to add rules to a connection matrix
;;; 2. Some functions
;;; 3. Connection-Matrix Object
;;; 4. Fuzzy-System Object
;;;

;;; [1] FUNCTION TO ADD RULES TO A CONNECTION MATRIX
;;;     (This function uses recursion)

define add_rules_to_CM(matrix,list_of_rules);
 lvars rule_number row column;

 if list_of_rules=[]
 then
 else
  hd(list_of_rules) -> rule_number;
  (rule_number * 2)-1 -> row;
  (rule_number * 2) -> column;

  1 -> matrix(row,column);
  add_rules_to_CM(matrix,tl(list_of_rules))
 endif;
enddefine;


;;; [2] SOME FUNCTIONS

define ismember(element,list);
 if list matches [== ^element ==]
  then true
  else false
 endif;
enddefine;

define subset_of(x,y);
 lvars x y check value;

 1 -> check;
 until x=[]
 do
  hd(x) -> value;
  if not(ismember(value,y))
  then
   0 -> check
  endif;
  tl(x) -> x
 enduntil;

 if check=1
  then true
  else false
 endif;
enddefine;


;;; [3] CONNECTION_MATRIX OBJECT

flavour CONNECTION_MATRIX;
 ivars matrix_name matrix list_of_rules credibility_weight;
 ivars history_of_decisions history_of_incs_decs;

 ;;; LIST OF METHODS
 ;;;
 ;;; 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; 2) save_to_file(x);

 ;;; METHOD: 1) create_a_connection_matrix(number_of_rules, rules);
 ;;; Method to create a connection matrix
 defmethod create_a_connection_matrix(number_of_rules rules);
  lvars number_of_rules rules size_of_matrix;
  rules -> list_of_rules;
  number_of_rules * 2 -> size_of_matrix;

  newarray([1 ^size_of_matrix 1 ^size_of_matrix],0) -> matrix;

  ;;; Add rules to the Connection Matrix (uses recursion)
  add_rules_to_CM(matrix,rules)
 enddefmethod;


;;; METHOD 2: save connection matrices to file
defmethod save_to_file(expert, structure);
 lvars expert structure filed;

 if expert = 1
 then
  if structure = 1
  then
   syscreate("mc1s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("mc1s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("mc1s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("mc1s4",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  endif;
 elseif expert = 2
 then
  if structure = 1
  then
   syscreate("mc2s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("mc2s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("mc2s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("mc2s4",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  endif;
 elseif expert = 3
 then
  if structure = 1
  then
   syscreate("mc3s1",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 2
  then
   syscreate("mc3s2",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  elseif structure = 3
  then
   syscreate("mc3s3",1,"line") -> filed;
   matrix -> datafile(filed);
   sysclose(filed);
  else
   syscreate("mc3s4",1,"line") -> filed;
```

```
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  elseif expert = 4
  then
    if structure = 1
    then
      syscreate("me4s1",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 2
    then
      syscreate("me4s2",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 3
    then
      syscreate("me4s3",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    else
      syscreate("me4s4",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  else
    if structure = 1
    then
      syscreate("me5s1",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 2
    then
      syscreate("me5s2",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    elseif structure = 3
    then
      syscreate("me5s3",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    else
      syscreate("me5s4",1,"line") -> filed;
      matrix -> datafile(filed);
      sysclose(filed);
    endif;
  endif;
enddefmethod;

endflavour;


;;; [4] FUZZY_SYSTEM OBJECT
;;;
;;;
;;;  LIST OF METHODS:
;;;
;;;    1) add_new_rules(s);
;;;    2) check_if_new_rules_already_exist(i);
;;;    3) add_rules_to_rulebase(s,e);
;;;    4) create_new_fuzzy_rules;

flavour FUZZY_SYSTEM;
 ivars type_of_structure fuzzy_variable thresholds number_of_potential_rules;
 ivars consequent_tally example_set_fuzzy_premise;
 ivars example_set_boolean_premise example_set_consequent;
 ivars number_of_fuzzy_premises number_of_boolean_premises;
 ivars number_of_consequences fuzzy_premises boolean_premises consequences;
 ivars fuzzy_peak_broadening fuzzy_asymmetry fuzzy_wedge_shaped;
 ivars fuzzy_visibility fuzzy_spacing_of_peaks fuzzy_plus_minus_asymmetry;
 ivars fuzzy_interference fuzzy_intensity_of_peak fuzzy_mismatch;
 ivars membership_functions potential_rules percent1 percent2 neg;
 ivars effectiveness no_of_rules;

 ;;; METHOD 1) add_new_rules(s);
 defmethod add_new_rules(s);
  lvars s value number_of_rules new_rules i j count last_value;
  lvars premise_number premise_value premise filed;

 if s=1
 then
  sysopen("noffp1",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_premises;
  sysclose(filed);
  sysopen("nofbp1",0,"line") -> filed;
  datafile(filed) -> number_of_boolean_premises;
  sysclose(filed);
  sysopen("noffc1",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("fp1",0,"line") -> filed;
  datafile(filed) -> fuzzy_premises;
  sysclose(filed);
  sysopen("bp1",0,"line") -> filed;
  datafile(filed) -> boolean_premises;
  sysclose(filed);
  sysopen("fc1",0,"line") -> filed;
  datafile(filed) -> consequences;
  sysclose(filed);
 elseif s=2
 then
  sysopen("noffp2",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_premises;
  sysclose(filed);
  sysopen("nofbp2",0,"line") -> filed;
  datafile(filed) -> number_of_boolean_premises;
  sysclose(filed);
  sysopen("noffc2",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("fp2",0,"line") -> filed;
  datafile(filed) -> fuzzy_premises;
  sysclose(filed);
  sysopen("bp2",0,"line") -> filed;
  datafile(filed) -> boolean_premises;
  sysclose(filed);
  sysopen("fc2",0,"line") -> filed;
  datafile(filed) -> consequences;
  sysclose(filed);
 elseif s=3
 then
  sysopen("noffp3",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_premises;
  sysclose(filed);
  sysopen("nofbp3",0,"line") -> filed;
  datafile(filed) -> number_of_boolean_premises;
  sysclose(filed);
  sysopen("noffc3",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("fp3",0,"line") -> filed;
  datafile(filed) -> fuzzy_premises;
  sysclose(filed);
  sysopen("bp3",0,"line") -> filed;
  datafile(filed) -> boolean_premises;
  sysclose(filed);
  sysopen("fc3",0,"line") -> filed;
  datafile(filed) -> consequences;
  sysclose(filed);
 else
  sysopen("noffp4",0,"line") -> filed;
  datafile(filed) -> number_of_fuzzy_premises;
  sysclose(filed);
  sysopen("nofbp4",0,"line") -> filed;
  datafile(filed) -> number_of_boolean_premises;
  sysclose(filed);
  sysopen("noffc4",0,"line") -> filed;
  datafile(filed) -> number_of_consequences;
  sysclose(filed);
  sysopen("fp4",0,"line") -> filed;
  datafile(filed) -> fuzzy_premises;
  sysclose(filed);
  sysopen("bp4",0,"line") -> filed;
  datafile(filed) -> boolean_premises;
  sysclose(filed);
  sysopen("fc4",0,"line") -> filed;
  datafile(filed) -> consequences;
  sysclose(filed);
 endif;

 nl(3);pr('          ADD NEW RULES TO FUZZY RULEBASE');nl(3);
 if s = 1
 then
  pr('Substrate Only Structure :');nl(2)
 elseif s=2
 then
  pr('Single Layer Structure :');nl(2)
 elseif s=3
```

```
then
 pr('Multiple_layers Structure :');nl(2)
elseif s=4
then
 pr('MQW Structure or Superlattice :');nl(2)
else
endif;

pr('How many new rules do you wish to add to the fuzzy rulebase  ');
readline() -> value;

hd(value) -> no_of_rules;
newarray([1 ^no_of_rules 1 4]) -> new_rules;
1 -> i;
until i > no_of_rules
do
 for j in [1 2 3 4]
 do
  [] -> new_rules(i,j);
 endfor;
 i + 1 -> i
enduntil;

1 -> count;
until count > no_of_rules
do
 [0] -> value;
 number_of_fuzzy_premises + 1 -> last_value;
 [] < > [^last_value] -> last_value;
 until value = last_value
 do
  nl(4);
  pr('CHOOSE A FUZZY PREMISE FOR THE NEW RULE :');nl(2);
  1 -> i;
  until i > number_of_fuzzy_premises
  do
   if i<10
   then
    pr('[');pr(i);pr(']  ')
   else
    pr('[');pr(i);pr('] ')
   endif;
   pr(fuzzy_premises(i));nl(1);
   i+1 -> i
  enduntil;
  if i<10
  then
   pr('[');pr(i);pr(']  ')
  else
   pr('[');pr(i);pr('] ')
  endif;
  pr('NO MORE FUZZY PREMISES');nl(3);

  pr('Please type the number of the FUZZY PREMISE ...');
  readline() -> value;
  hd(value) -> premise_number;

  if not(value = last_value)
  then
   nl(3);pr('PLEASE CHOOSE A VALUE FOR THIS PREMISE :');nl(2);
   pr('[1]  EXTREME');nl(1);
   pr('[2]  VERY');nl(1);
   pr('[3]  FAIRLY');nl(1);
   pr('[4]  SOME');nl(1);
   pr('[5]  NONE');nl(3);
   pr('Please type the number of the VALUE OF THE PREMISE ...');
   readline() -> value;
   nl(2);
   hd(value) -> premise_value
  endif;
  pr('_____');nl(3);
  (premise_number - 1) * 5 + premise_value -> premise;
  new_rules(count,1) < > [^premise] -> new_rules(count,1)
 enduntil;

 [0] -> value;
 number_of_boolean_premises + 1 -> last_value;
 [] < > [^last_value] -> last_value;
 until value = last_value
 do
  pr('CHOOSE A BOOLEAN PREMISE FOR THE NEW RULE :');nl(2);
  1 -> i;
  until i > number_of_boolean_premises
  do
   if i<10
   then
    pr('[');pr(i);pr(']  ')
   else
    pr('[');pr(i);pr('] ')
   endif;
   pr(boolean_premises(i));nl(1);
   i+1 -> i
  enduntil;
  if i<10
  then
   pr('[');pr(i);pr(']  ')
  else
   pr('[');pr(i);pr('] ')
  endif;
  pr('NO MORE BOOLEAN PREMISES');nl(3);

  pr('Please type the number of the BOOLEAN PREMISE ...');readline()
                -> value;
  hd(value) -> premise_number;

  if not(value = last_value)
  then
   nl(3);pr('PLEASE CHOOSE A VALUE FOR THIS PREMISE :');nl(2);
   pr('[1]  TRUE');nl(1);
   pr('[2]  FALSE');nl(3);
   pr('Please type the number of the value ...');readline() -> value;
   nl(2);
   hd(value) -> premise_value
  endif;
  pr('_____');nl(3);
  (premise_number - 1) * 2 + premise_value -> premise;
  new_rules(count,2) < > [^premise] -> new_rules(count,2)
 enduntil;

 [0] -> value;
 [1] -> last_value;
 until value = last_value
 do
  pr('CHOOSE A FUZZY CONSEQUENCE FOR THE NEW RULE :');
  nl(2);
  1 -> i;
  until i > number_of_consequences
  do
   if i<10
   then
    pr('[');pr(i);pr(']  ')
   else
    pr('[');pr(i);pr('] ')
   endif;
   pr(consequences(i));nl(1);
   i+1 -> i
  enduntil;

  nl(2);
  pr('Please type the number of the FUZZY CONSEQUENCE ...');
  readline() -> value;
  hd(value) -> premise_number;

  if not(value = last_value)
  then
   nl(3);
   pr('PLEASE CHOOSE A VALUE FOR THIS CONSEQUENCE :');
   nl(2);
   pr('[1]  EXTREME');nl(1);
   pr('[2]  VERY');nl(1);
   pr('[3]  FAIRLY');nl(1);
   pr('[4]  SOME');nl(1);
   pr('[5]  NONE');nl(3);
   pr('Please type the number of the VALUE OF
                THE CONSEQUENCE ...');
   readline() -> value;
   nl(2);
   hd(value) -> premise_value
  endif;
  pr('_____');nl(3);
  new_rules(count,3) < > [^premise_value] -> new_rules(count,3);
  new_rules(count,4) < > [^premise_number] -> new_rules(count,4);
  last_value -> value;
 enduntil;
 count + 1 -> count
enduntil;
```

```
  if s=1
  then
    syscreate("potrule1",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif s=2
  then
    syscreate("potrule2",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
    syscreate("some1",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif s=3
  then
    syscreate("potrule3",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  elseif s=4
  then
    syscreate("potrule4",1,"line") -> filed;
    potential_rules -> datafile(filed);
    sysclose(filed);
  else
  endif;
enddefmethod;


;;; METHOD 2) check_if_new_rules_already_exist(i);
defmethod check_if_new_rules_already_exist(i);
  lvars filed i number_of_rules;
  lvars existing_rules potential_rules rule_number consequent;

    if i = 1
    then
      sysopen("rules1",0,"line") -> filed;
      datafile(filed) -> existing_rules;
      sysclose(filed);
      sysopen("rno1",0,"line") -> filed;
      datafile(filed) -> number_of_rules;
      sysclose(filed);
      sysopen("potrule1",0,"line") -> filed;
      datafile(filed) -> potential_rules;
      sysclose(filed);
    elseif i = 2
    then
      sysopen("rules2",0,"line") -> filed;
      datafile(filed) -> existing_rules;
      sysclose(filed);
      sysopen("rno2",0,"line") -> filed;
      datafile(filed) -> number_of_rules;
      sysclose(filed);
      sysopen("potrule2",0,"line") -> filed;
      datafile(filed) -> potential_rules;
      sysclose(filed);
    elseif i = 3
    then
      sysopen("rules3",0,"line") -> filed;
      datafile(filed) -> existing_rules;
      sysclose(filed);
      sysopen("rno3",0,"line") -> filed;
      datafile(filed) -> number_of_rules;
      sysclose(filed);
      sysopen("potrule3",0,"line") -> filed;
      datafile(filed) -> potential_rules;
      sysclose(filed);
    else
      sysopen("rules4",0,"line") -> filed;
      datafile(filed) -> existing_rules;
      sysclose(filed);
      sysopen("rno4",0,"line") -> filed;
      datafile(filed) -> number_of_rules;
      sysclose(filed);
      sysopen("potrule4",0,"line") -> filed;
      datafile(filed) -> potential_rules;
      sysclose(filed);
    endif;

    1 -> rule_number;
    until rule_number > number_of_rules
    do
      existing_rules(rule_number,4) -> consequent;
      if consequent = []
      then
        hd(consequent) -> consequent;

        if existing_rules(rule_number,1) = potential_rules(consequent,1) and
          existing_rules(rule_number,2) = potential_rules(consequent,2) and
          existing_rules(rule_number,3) = potential_rules(consequent,3) and
          existing_rules(rule_number,4) = potential_rules(consequent,4)
        then
          [] -> potential_rules(consequent,1);
          [] -> potential_rules(consequent,2);
          [] -> potential_rules(consequent,3);
          [] -> potential_rules(consequent,4)
        endif;

        if subset_of(existing_rules(rule_number,1),
              potential_rules(consequent,1)) or
          subset_of(existing_rules(rule_number,2),
              potential_rules(consequent,2))
        then
          [] -> potential_rules(consequent,1);
          [] -> potential_rules(consequent,2);
          [] -> potential_rules(consequent,3);
          [] -> potential_rules(consequent,4)
        endif;
      endif;
      rule_number + 1 -> rule_number
    enduntil;

    if i = 1
    then
      syscreate("potrule1",1,"line") -> filed;
      potential_rules -> datafile(filed);
      sysclose(filed);
    elseif i = 2
    then
      syscreate("potrule2",1,"line") -> filed;
      potential_rules -> datafile(filed);
      sysclose(filed);
      syscreate("some2",1,"line") -> filed;
      potential_rules -> datafile(filed);
      sysclose(filed);
    elseif i = 3
    then
      syscreate("potrule3",1,"line") -> filed;
      potential_rules -> datafile(filed);
      sysclose(filed);
    elseif i = 4
    then
      syscreate("potrule4",1,"line") -> filed;
      potential_rules -> datafile(filed);
      sysclose(filed);
    else
    endif;
enddefmethod;


;;; METHOD 3) add_rules_to_rulebase(s,e);
defmethod add_rules_to_rulebase(s,e);
  lvars s i e filed new_rules existing_rules;
  lvars number_of_consequences number_of_consequents;
  lvars new_ruleset number_of_rules new_number_of_rules;
  lvars consequent list_of_rules value value2 count count2;
  lvars number_of_experts connection_matrix rules_for_substrate_only;
  lvars rules_for_single_layer rules_for_multiple_layers rules_for_MQW;
  lvars list_of_fuzzy_variables number_of_fuzzy_variables;
  lvars new_number_of_fuzzy_variables temp_list rule_history r_hist;
  lvars masking_of_rule mask_rule fuzzy_rules mf mf2 layers variable_history;
  lvars var_hist var_changes var_ch;

  sysopen("noexp",0,"line") -> filed;
  datafile(filed) -> number_of_experts;
  sysclose(filed);

  make_instance([CONNECTION_MATRIX]) -> connection_matrix;
    if s = 1
    then
      sysopen("noffc1",0,"line") -> filed;
      datafile(filed) -> number_of_consequences;
      sysclose(filed);
      sysopen("potrule1",0,"line") -> filed;
      datafile(filed) -> new_rules;
      sysclose(filed);
      sysopen("rno1",0,"line") -> filed;
      datafile(filed) -> number_of_rules;
```

```
    sysclose(filed);
    sysopen("rules1",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    if e = 1
    then
      sysopen("lofr1s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 2
    then
      sysopen("lofr2s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 3
    then
      sysopen("lofr3s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 4
    then
      sysopen("lofr4s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    else
      sysopen("lofr5s1",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    endif;
  elseif s = 2
  then
    sysopen("noffc2",0,"line") -> filed;
    datafile(filed) -> number_of_consequences;
    sysclose(filed);
    sysopen("potrule2",0,"line") -> filed;
    datafile(filed) -> new_rules;
    sysclose(filed);
    sysopen("rno2",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("rules2",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    if e = 1
    then
      sysopen("lofr1s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 2
    then
      sysopen("lofr2s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 3
    then
      sysopen("lofr3s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 4
    then
      sysopen("lofr4s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    else
      sysopen("lofr5s2",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    endif;
  elseif s = 3
  then
    sysopen("noffc3",0,"line") -> filed;
    datafile(filed) -> number_of_consequences;
    sysclose(filed);
    sysopen("potrule3",0,"line") -> filed;
    datafile(filed) -> new_rules;
    sysclose(filed);
    sysopen("rno3",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("rules3",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    if e = 1
    then
      sysopen("lofr1s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 2
    then
      sysopen("lofr2s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 3
    then
      sysopen("lofr3s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 4
    then
      sysopen("lofr4s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    else
      sysopen("lofr5s3",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    endif;
  elseif s = 4
  then
    sysopen("noffc4",0,"line") -> filed;
    datafile(filed) -> number_of_consequences;
    sysclose(filed);
    sysopen("potrule4",0,"line") -> filed;
    datafile(filed) -> new_rules;
    sysclose(filed);
    sysopen("rno4",0,"line") -> filed;
    datafile(filed) -> number_of_rules;
    sysclose(filed);
    sysopen("rules4",0,"line") -> filed;
    datafile(filed) -> existing_rules;
    sysclose(filed);
    if e = 1
    then
      sysopen("lofr1s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 2
    then
      sysopen("lofr2s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 3
    then
      sysopen("lofr3s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    elseif e = 4
    then
      sysopen("lofr4s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    else
      sysopen("lofr5s4",0,"line") -> filed;
      datafile(filed) -> list_of_rules;
      sysclose(filed);
    endif;
  else
  endif;

  number_of_consequences * 5 -> number_of_consequents;

  ;;; Calculate the total number of rules when new rules are added
  number_of_rules -> new_number_of_rules;
  1 -> consequent;
  until consequent > no_of_rules
  do
    if not(new_rules(consequent,1) = [] and new_rules(consequent,2) = [])
    then
      new_number_of_rules + 1 -> new_number_of_rules
    endif;
    consequent + 1 -> consequent
  enduntil;

  ;;; Add the existing rules to the new ruleset
  newarray([1 ^new_number_of_rules 1 4]) -> new_ruleset;
  1 -> i;
  until i > number_of_rules
  do
```

```
  existing_rules(i,1) -> new_ruleset(i,1);
  existing_rules(i,2) -> new_ruleset(i,2);
  existing_rules(i,3) -> new_ruleset(i,3);
  existing_rules(i,4) -> new_ruleset(i,4);
  i+1 -> i
enduntil;

[] -> temp_list;
[] -> list_of_fuzzy_variables;
;;; Add new rules into the new ruleset
1 -> consequent;
until consequent > no_of_rules
do
 if not(new_rules(consequent,1) = [] and new_rules(consequent,2) = [])
 then
  new_rules(consequent,1) -> new_ruleset(i,1);
  new_rules(consequent,2) -> new_ruleset(i,2);
  new_rules(consequent,3) -> new_ruleset(i,3);
  new_rules(consequent,4) -> new_ruleset(i,4);
  ;;; Make a list the numbers of fuzzy premises in each new rule
  length(new_rules(consequent,1)) -> value;
  temp_list <> [^value] -> temp_list;
  ;;; Record the fuzzy premises in a list
  new_rules(consequent,1) -> value2;
  list_of_fuzzy_variables <> [^^value2] -> list_of_fuzzy_variables;
  list_of_rules <> [^i] -> list_of_rules;
  i+1 -> i
 endif;
 consequent + 1 -> consequent
enduntil;

if s = 1
then
 sysopen("rhist1",0,"line") -> filed;
 datafile(filed) -> rule_history;
 sysclose(filed);
 sysopen("rmask1",0,"line") -> filed;
 datafile(filed) -> masking_of_rule;
 sysclose(filed);
 sysopen("fuzzno1",0,"line") -> filed;
 datafile(filed) -> number_of_fuzzy_variables;
 sysclose(filed);
 sysopen("frules1",0,"line") -> filed;
 datafile(filed) -> fuzzy_rules;
 sysclose(filed);
 sysopen("varhist1",0,"line") -> filed;
 datafile(filed) -> variable_history;
 sysclose(filed);
 sysopen("varch1",0,"line") -> filed;
 datafile(filed) -> var_changes;
 sysclose(filed);
 sysopen("memf1",0,"line") -> filed;
 datafile(filed) -> mf;
 sysclose(filed);
 sysopen("l1",0,"line") -> filed;
 datafile(filed) -> layers;
 sysclose(filed);

 ;;; Substrate Only
 sysopen("lofr1s1",0,"line") -> filed;
 datafile(filed) -> rules_for_substrate_only;
 sysclose(filed);
 if e = 1
 then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
   rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
   count + 1 -> count
  enduntil;
 endif;
 syscreate("lofr1s1",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
 connection_matrix <- save_to_file(1,1);

 sysopen("lofr2s1",0,"line") -> filed;
 datafile(filed) -> rules_for_substrate_only;
 sysclose(filed);
 if e = 2
 then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
   rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
   count + 1 -> count
  enduntil;
 endif;
 syscreate("lofr2s1",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
 connection_matrix <- save_to_file(2,1);

 sysopen("lofr3s1",0,"line") -> filed;
 datafile(filed) -> rules_for_substrate_only;
 sysclose(filed);
 if e = 3
 then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
   rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
   count + 1 -> count
  enduntil;
 endif;
 syscreate("lofr3s1",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
 connection_matrix <- save_to_file(3,1);

 sysopen("lofr4s1",0,"line") -> filed;
 datafile(filed) -> rules_for_substrate_only;
 sysclose(filed);
 if e = 4
 then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
   rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
   count + 1 -> count
  enduntil;
 endif;
 syscreate("lofr4s1",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
 connection_matrix <- save_to_file(4,1);
 if e = 5
 then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
   rules_for_substrate_only <> [^count] -> rules_for_substrate_only;
   count + 1 -> count
  enduntil;
 endif;
 syscreate("lofr5s1",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_substrate_only);
 connection_matrix <- save_to_file(5,1);

elseif s = 2
then
 sysopen("rhist2",0,"line") -> filed;
 datafile(filed) -> rule_history;
 sysclose(filed);
 sysopen("rmask2",0,"line") -> filed;
 datafile(filed) -> masking_of_rule;
 sysclose(filed);
 sysopen("fuzzno2",0,"line") -> filed;
 datafile(filed) -> number_of_fuzzy_variables;
 sysclose(filed);
 sysopen("frules2",0,"line") -> filed;
 datafile(filed) -> fuzzy_rules;
 sysclose(filed);
 sysopen("varhist2",0,"line") -> filed;
 datafile(filed) -> variable_history;
 sysclose(filed);
 sysopen("varch2",0,"line") -> filed;
```

```
datafile(filed) -> var_changes;
sysclose(filed);
sysopen("memf2",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);
sysopen("l2",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);

;;; Single Layer
sysopen("lofr1s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 1
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer <> [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr1s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_single_layer);
connection_matrix <- save_to_file(1,2);

sysopen("lofr2s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 2
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer <> [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr2s2",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_single_layer);
connection_matrix <- save_to_file(2,2);

sysopen("lofr3s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 3
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer <> [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr3s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_single_layer);
connection_matrix <- save_to_file(3,2);

sysopen("lofr4s2",0,"line") -> filed;
datafile(filed) -> rules_for_single_layer;
sysclose(filed);
if e = 4
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer <> [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr4s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_single_layer);
connection_matrix <- save_to_file(4,2);
if e = 5
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_single_layer <> [^count] -> rules_for_single_layer;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr5s2",1,"line") -> filed;
rules_for_single_layer -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_single_layer);
connection_matrix <- save_to_file(5,2);

elseif s = 3
then
sysopen("rhist3",0,"line") -> filed;
datafile(filed) -> rule_history;
sysclose(filed);
sysopen("rmask3",0,"line") -> filed;
datafile(filed) -> masking_of_rule;
sysclose(filed);
sysopen("fuzzno3",0,"line") -> filed;
datafile(filed) -> number_of_fuzzy_variables;
sysclose(filed);
sysopen("frules3",0,"line") -> filed;
datafile(filed) -> fuzzy_rules;
sysclose(filed);
sysopen("varhist3",0,"line") -> filed;
datafile(filed) -> variable_history;
sysclose(filed);
sysopen("varch3",0,"line") -> filed;
datafile(filed) -> var_changes;
sysclose(filed);
sysopen("memf3",0,"line") -> filed;
datafile(filed) -> mf;
sysclose(filed);
sysopen("l3",0,"line") -> filed;
datafile(filed) -> layers;
sysclose(filed);

;;; Multiple Layers
sysopen("lofr1s3",0,"line") -> filed;
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 1
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers <> [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr1s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(1,3);

sysopen("lofr2s3",0,"line") -> filed;
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 2
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers <> [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr2s3",1,"line") -> filed;
rules_for_substrate_only -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(2,3);

sysopen("lofr3s3",0,"line") -> filed;
```

```
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 3
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr3s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(3,3);

sysopen("lofr4s3",0,"line") -> filed;
datafile(filed) -> rules_for_multiple_layers;
sysclose(filed);
if e = 4
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr4s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(4,3);
if e = 5
then
  number_of_rules + 1 -> count;
  until count > new_number_of_rules
  do
    rules_for_multiple_layers < > [^count] -> rules_for_multiple_layers;
    count + 1 -> count
  enduntil;
endif;
syscreate("lofr5s3",1,"line") -> filed;
rules_for_multiple_layers -> datafile(filed);
sysclose(filed);
connection_matrix<-create_a_connection_matrix(new_number_of_rules,
      rules_for_multiple_layers);
connection_matrix <- save_to_file(5,3);

elseif s = 4
then
 sysopen("rhist4",0,"line") -> filed;
 datafile(filed) -> rule_history;
 sysclose(filed);
 sysopen("rmask4",0,"line") -> filed;
 datafile(filed) -> masking_of_rule;
 sysclose(filed);
 sysopen("fuzzno4",0,"line") -> filed;
 datafile(filed) -> number_of_fuzzy_variables;
 sysclose(filed);
 sysopen("frules4",0,"line") -> filed;
 datafile(filed) -> fuzzy_rules;
 sysclose(filed);
 sysopen("varhist4",0,"line") -> filed;
 datafile(filed) -> variable_history;
 sysclose(filed);
 sysopen("varch4",0,"line") -> filed;
 datafile(filed) -> var_changes;
 sysclose(filed);
 sysopen("mcmf4",0,"line") -> filed;
 datafile(filed) -> mf;
 sysclose(filed);
 sysopen("l4",0,"line") -> filed;
 datafile(filed) -> layers;
 sysclose(filed);

 ;;; MQW Structure and Superlattices
 sysopen("lofr1s4",0,"line") -> filed;
 datafile(filed) -> rules_for_MQW;
 sysclose(filed);
 if e = 1
 then
   number_of_rules + 1 -> count;
   until count > new_number_of_rules
   do
     rules_for_MQW < > [^count] -> rules_for_MQW;
     count + 1 -> count
   enduntil;
 endif;
 syscreate("lofr1s4",1,"line") -> filed;
 rules_for_MQW -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
       rules_for_MQW);
 connection_matrix <- save_to_file(1,4);

 sysopen("lofr2s4",0,"line") -> filed;
 datafile(filed) -> rules_for_MQW;
 sysclose(filed);
 if e = 2
 then
   number_of_rules + 1 -> count;
   until count > new_number_of_rules
   do
     rules_for_MQW < > [^count] -> rules_for_MQW;
     count + 1 -> count
   enduntil;
 endif;
 syscreate("lofr2s4",1,"line") -> filed;
 rules_for_substrate_only -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
       rules_for_MQW);
 connection_matrix <- save_to_file(2,4);

 sysopen("lofr3s4",0,"line") -> filed;
 datafile(filed) -> rules_for_MQW;
 sysclose(filed);
 if e = 3
 then
   number_of_rules + 1 -> count;
   until count > new_number_of_rules
   do
     rules_for_MQW < > [^count] -> rules_for_MQW;
     count + 1 -> count
   enduntil;
 endif;
 syscreate("lofr3s4",1,"line") -> filed;
 rules_for_MQW -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
       rules_for_MQW);
 connection_matrix <- save_to_file(3,4);

 sysopen("lofr4s4",0,"line") -> filed;
 datafile(filed) -> rules_for_MQW;
 sysclose(filed);
 if e = 4
 then
   number_of_rules + 1 -> count;
   until count > new_number_of_rules
   do
     rules_for_MQW < > [^count] -> rules_for_MQW;
     count + 1 -> count
   enduntil;
 endif;
 syscreate("lofr4s4",1,"line") -> filed;
 rules_for_MQW -> datafile(filed);
 sysclose(filed);
 connection_matrix<-create_a_connection_matrix(new_number_of_rules,
       rules_for_MQW);
 connection_matrix <- save_to_file(4,4);
 if e = 5
 then
   number_of_rules + 1 -> count;
   until count > new_number_of_rules
   do
     rules_for_MQW < > [^count] -> rules_for_MQW;
     count + 1 -> count
   enduntil;
 endif;
 syscreate("lofr5s4",1,"line") -> filed;
 rules_for_MQW -> datafile(filed);
 sysclose(filed);
```

```
  connection_matrix <-create_a_connection_matrix(new_number_of_rules,
        rules_for_MQW);
  connection_matrix <- save_to_file(5,4);
 else
 endif;

 newarray([1 ^new_number_of_rules]) -> r_hist;
 newarray([1 ^new_number_of_rules]) -> mask_rule;
 1 -> i;
 until i > number_of_rules
 do
  rule_history(i) -> r_hist(i);
  masking_of_rule(i) -> mask_rule(i);
  i+1 -> i
 enduntil;
 until i > new_number_of_rules
 do
  [] -> r_hist(i);
  "ASSERT" -> mask_rule(i);
  i+1 -> i
 enduntil;

 number_of_fuzzy_variables -> new_number_of_fuzzy_variables;
 number_of_rules + 1 -> count;
 until count > new_number_of_rules
 do
  hd(temp_list) -> value;
  fuzzy_rules(value,1) <> [^count] -> fuzzy_rules(value,1);
  1 -> count2;
  until count2 > value
  do
   new_number_of_fuzzy_variables + 1
              -> new_number_of_fuzzy_variables;
   fuzzy_rules(value,2) <> [^new_number_of_fuzzy_variables]
              -> fuzzy_rules(value,2);
   count2 + 1 -> count2
  enduntil;
  count + 1 -> count;
  tl(temp_list) -> temp_list
 enduntil;

 newarray([1 2 1 ^new_number_of_fuzzy_variables]) -> var_hist;
 newarray([1 ^new_number_of_fuzzy_variables]) -> var_ch;
 newarray([1 ^new_number_of_fuzzy_variables]) -> mf2;
 1 -> i;
 until i > number_of_fuzzy_variables
 do
  variable_history(1,i) -> var_hist(1,i);
  variable_history(2,i) -> var_hist(2,i);
  var_changes(i) -> var_ch(i);
  mf(i) -> mf2(i);
  i+1 -> i
 enduntil;
 until list_of_fuzzy_variables = []
 do
  [] -> var_hist(1,i);
  [] -> var_hist(2,i);
  [] -> var_ch(i);
  hd(list_of_fuzzy_variables) -> value;
  1 -> count;
  until value < 6
  do
   value - 5 -> value;
   count + 1 -> count;
  enduntil;
  count -> mf2(i);
  layers <> [1] -> layers;
  i+1 ->i;
  tl(list_of_fuzzy_variables) -> list_of_fuzzy_variables
 enduntil;

 if s = 1
 then
  syscreate("rno1",1,"line") -> filed;
  new_number_of_rules -> datafile(filed);
  sysclose(filed);
  syscreate("rules1",1,"line") -> filed;
  new_rules -> datafile(filed);
  sysclose(filed);
  if e=1
  then
   syscreate("lofr1s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=2
  then
   syscreate("lofr2s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=3
  then
   syscreate("lofr3s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=4
  then
   syscreate("lofr4s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=5
  then
   syscreate("lofr5s1",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  else
  endif;
  syscreate("rhist1",1,"line") -> filed;
  r_hist -> datafile(filed);
  sysclose(filed);
  syscreate("rmask1",1,"line") -> filed;
  mask_rule -> datafile(filed);
  sysclose(filed);
  syscreate("fuzzno1",1,"line") -> filed;
  new_number_of_fuzzy_variables -> datafile(filed);
  sysclose(filed);
  syscreate("frules1",1,"line") -> filed;
  fuzzy_rules -> datafile(filed);
  sysclose(filed);
  syscreate("varhist1",1,"line") -> filed;
  var_hist -> datafile(filed);
  sysclose(filed);
  syscreate("varch1",1,"line") -> filed;
  var_ch -> datafile(filed);
  sysclose(filed);
  syscreate("memf1",1,"line") -> filed;
  mf2 -> datafile(filed);
  sysclose(filed);
  syscreate("l1",1,"line") -> filed;
  layers -> datafile(filed);
  sysclose(filed);
 elseif s = 2
 then
  syscreate("rno2",1,"line") -> filed;
  new_number_of_rules -> datafile(filed);
  sysclose(filed);
  syscreate("rules2",1,"line") -> filed;
  new_rules -> datafile(filed);
  sysclose(filed);
  if e=1
  then
   syscreate("lofr1s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=2
  then
   syscreate("lofr2s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=3
  then
   syscreate("lofr3s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=4
  then
   syscreate("lofr4s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  elseif e=5
  then
   syscreate("lofr5s2",1,"line") -> filed;
   list_of_rules -> datafile(filed);
   sysclose(filed);
  else
  endif;
  syscreate("rhist2",1,"line") -> filed;
  r_hist -> datafile(filed);
  sysclose(filed);
```

```
    syscreate("rmask2",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno2",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules2",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist2",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch2",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf2",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l2",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  elseif s = 3
  then
    syscreate("rno3",1,"line") -> filed;
    new_number_of_rules -> datafile(filed);
    sysclose(filed);
    syscreate("rules3",1,"line") -> filed;
    new_rules -> datafile(filed);
    sysclose(filed);
    if e=1
    then
      syscreate("lofr1s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=2
    then
      syscreate("lofr2s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=3
    then
      syscreate("lofr3s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=4
    then
      syscreate("lofr4s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=5
    then
      syscreate("lofr5s3",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    else
    endif;
    syscreate("rhist3",1,"line") -> filed;
    r_hist -> datafile(filed);
    sysclose(filed);
    syscreate("rmask3",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno3",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules3",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist3",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch3",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf3",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l3",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  elseif s = 4
  then
    syscreate("rno4",1,"line") -> filed;
    new_number_of_rules -> datafile(filed);
    sysclose(filed);
    syscreate("rules4",1,"line") -> filed;
    new_rules -> datafile(filed);
    sysclose(filed);
    if e=1
    then
      syscreate("lofr1s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=2
    then
      syscreate("lofr2s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=3
    then
      syscreate("lofr3s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=4
    then
      syscreate("lofr4s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    elseif e=5
    then
      syscreate("lofr5s4",1,"line") -> filed;
      list_of_rules -> datafile(filed);
      sysclose(filed);
    else
    endif;
    syscreate("rhist4",1,"line") -> filed;
    r_hist -> datafile(filed);
    sysclose(filed);
    syscreate("rmask4",1,"line") -> filed;
    mask_rule -> datafile(filed);
    sysclose(filed);
    syscreate("fuzzno4",1,"line") -> filed;
    new_number_of_fuzzy_variables -> datafile(filed);
    sysclose(filed);
    syscreate("frules4",1,"line") -> filed;
    fuzzy_rules -> datafile(filed);
    sysclose(filed);
    syscreate("varhist4",1,"line") -> filed;
    var_hist -> datafile(filed);
    sysclose(filed);
    syscreate("varch4",1,"line") -> filed;
    var_ch -> datafile(filed);
    sysclose(filed);
    syscreate("memf4",1,"line") -> filed;
    mf2 -> datafile(filed);
    sysclose(filed);
    syscreate("l4",1,"line") -> filed;
    layers -> datafile(filed);
    sysclose(filed);
  else
  endif;
enddefmethod;


;;; METHOD 4) create_new_fuzzy_rules;
defmethod create_new_fuzzy_rules;
lvars value expert continue structure;

 nl(3);pr('   ADDING NEW RULES :');nl(2);
 [] -> value;
 until value = [1] or value = [2] or value = [3] or value = [4] or
         value = [5]
 do
  pr('Which expert is adding the rules:');nl(2);
  pr('  1.  Professor B. K. Tanner');nl(1);
  pr('  2.  Professor D. K. Bowen');nl(1);
  pr('  3.  Doctor N. Lowley');nl(1);
  pr('  4.  Doctor C. R. Thomas');nl(1);
  pr('  5.  DUMMY expert');nl(2);
  pr('Please type the number of your choice ...');readline() -> value;
  value -> expert;
 enduntil;

 [Y] -> continue;

 while continue = [Y] or continue = [y]
 do
  [] -> value;
```

```
    until value = [1] or value = [2] or value = [3] or value = [4]
    do
      nl(4);pr('For which type of structure are you entering new rules:');nl(2);
      pr('   1.  Substrate Only Structure');nl(1);
      pr('   2.  Single Layer Structure');nl(1);
      pr('   3.  Multiple Layers Structure');nl(1);
      pr('   4.  MQW Structure or Superlattice');nl(2);
      pr('Please type the number of your choice ...');readline() -> value;

      value -> structure;
    enduntil;

    hd(structure) -> structure;

    ^add_new_rules(structure);
    ^check_if_new_rules_already_exist(structure);
    ^add_rules_to_rulebase(structure,expert);

    nl(3);pr('DISPLAY NEW RULE FOR THE USER TO
              ACCEPT OR REJECT');nl(2);
    pr('Press enter to continue ...');readline()->value;

    nl(3);pr('Do you wish to add rules for a different structure (Y/N) ? ');
    readline() -> value;
    value -> continue;
  endwhile;
 enddefmethod;

endflavour;
```

APPENDIX E

External data files used by the fuzzy system.

In Table 1, there is a separate file for each partition of the rulebase.

In Table 2, there is only one data file.

TABLE 1:

Name of file	Contents
rno1, rno2, rno3, rno4	number of rules
rules1, rules2, rules3, rules4	ruleset stored as an array of numbered representations of rule elements
nofp1, nofp2, nofp3, nofp4	number of fuzzy premises
nofbp1, nofbp2, nofbp3, nofbp4	number of Boolean premises
noffc1, noffc2, noffc3, noffc4	number of fuzzy consequences
fp1, fp2, fp3, fp4	names of fuzzy premises
bp1, bp2, bp3, bp4	names of Boolean premises
fc1, fc2, fc3, fc4	names of fuzzy consequences
angst1, angst2, angst3, angst4	control variable used to decide whether to update connection matrices
rhist1, rhist2, rhist3, rhist4	history of decisions for each rule
rmask1, rmask2, rmask3, rmask4	switches used to decide whether rules are Asserted or Unasserted
fuzzno1, fuzzno2, fuzzno3, fuzzno4	number of fuzzy variables
memfunc1, memfunc2, memfunc3, memfunc4	membership functions for each fuzzy variable
varhist1, varhist2, varhist3, varhist4	history of values for fuzzy premise variables
varch1, varch2, varch3, varch4	history of changes made to the membership functions of fuzzy premises
frules1, frules2, frules3, frules4	lists linking fuzzy variables with their associated rules
memf1, memf2, memf3, memf4	values associating fuzzy premises with membership functions
l1, l2, l3, l4	layer numbers for each rule
rvar1, rvar2, rvar3, rvar4	linguistic variables used in each rule
potrule1, potrule2, potrule3, potrule4	potential rules created by induction, initialised to empty
conrec1, conrec2, conrec3, conrec4	tally of occurrences of each fuzzy consequent in the example set
pexset1, pexset2, pexset3, pexset4	example set for fuzzy premises used in inductive learning
bexset1, bexset2, bexset3, bexset4	example set of Boolean premises used in inductive learning
cexset1, cexset2, cexset3, cexset4	example set of fuzzy consequents used in inductive learning
cinr1, cinr2, cinr3, cinr4	number of occurrences of fuzzy consequents in existing rules

Table 2:

Name of file	Contents
noexp	number of experts (including a dummy expert)
$lofr_i{}^s{}_j$	list of rules for expert i and partition (or structure) j
$mc_i{}^s{}_j$	connection matrix for expert i and partition (or structure) j
credwt	credibility weights of experts
histdec	history of decisions for each expert
histid	history of increments/decrements to credibility weight of each expert
pfors	percentage threshold for success
pforf	percentage threshold for failure
mathn	value of N in Equation (4.1)
mathro	value for σ in Equation (4.1)
mathk	value for κ in Equation (4.1)
histrv	history of outcome of decision O values
pfunc	history of p values in Equation (4.1)
qfunc	history of q values in Equation (4.1)
ffunc	history of values for function f in Equation (4.1)
incvals	history of inc and dec values calculated using Equation (4.1)
pthresh	positive threshold for O used to adapt the fuzzy system

APPENDIX: F

Example session using fuzzy system for X-ray rocking curve analysis.

Question and Answer Session for a Single Layer Experimental Rocking Curve of layer AlGaAs on a GaAs substrate:

```
PLEASE DESCRIBE THE EXPERIMENTAL ROCKING CURVE ...

            LEADING QUESTION

Which of the following best describes the structure?

    (a) substrate only
    (b) a single layer
    (c) MQW structure
    (d) non-graded multiple layers
    (e) superlattice capped with a few layers top and bottom;
    (f) a single graded layer
    (g) graded multiple layers
    (h) graded superlattice.

Type the letter of your choice ... ? b

            A SINGLE LAYER STRUCTURE

            GENERAL INFORMATION ON THE EXPERIMMENT

1. What is the Wavelength of the X-ray beam (in angstroms) ? 1.541

2. Please enter the reflection indices for this experiment,
   i.e. the h,k,l Miller indices multiplied by the order of diffraction:

      H:  ? 0

      K:  ? 0

      L:  ? 4

3. Which of the following is the substrate material?

    (a) Si
    (b) Ge
    (c) AlP
    (d) AlAs
    (e) AlSb
    (f) GaP
    (g) GaAs
    (h) GaSb
    (i) InP
    (j) InAs
    (k) InSb

Type the letter of your choice ... ? g

4. Which of the following is the reflection orientation?

    (a) Symmetric
    (b) Asymmetric with glancing incidence
    (c) Asymmetric with glancing exit

Type the letter of your choice ... ? a

5. Please enter the surface normal indices for this experiment,

      H:  ? 0

      K:  ? 0

      L:  ? 1

6. Please type in the scan range of your rocking curve (in arc secs):

       Start :  ? -500

       Finish:  ? 500

            DESCRIPTION OF LAYER

(a) Which of the following is the layer material?

  ENDMEMBERS:
   (a) Si                  (g) GaAs
   (b) Ge                  (h) GaSb
   (c) AlP                 (i) InP
   (d) AlAs                (j) InAs
   (e) AlSb                (k) InSb
   (f) GaP

  COMPOUNDS:
   (l) GeSi                (q) InAsP
   (m) InGaP               (r) GaAsP
   (n) InGaAs              (s) InGaAlAs
   (o) InGaSb              (t) GaInAsP
   (p) AlGaAs

  Type the letter of your choice ... ? p
```

(b) What is the estimated thickness of the layer (in microns) ? 1.45

DESCRIPTION OF THE CURVE

1. How many peaks are there in the rocking curve?

(a) one peak;
(b) two peaks;
(c) more than two peaks;
(d) no peaks whatsoever.

Type the letter of your choice ... ? b

2. Does the layer peak overlap with the substrate peak (Y/N) ? n

2(a). Type in a measure of PEAK SEPARATION between the substrate and layer peaks (in arc seconds): ? 198

3. Is there a possibility of a RELAXED layer at the interface (Y/N)? n

DESCRIPTION OF THE SUBSTRATE PEAK

1. Please type in the Full Width at Half Maximum(FWHM) of the substrate peak (in arc seconds) ? 6.55

2. What is the integrated intensity of the substrate peak ? 70000

3. How ASYMMETRIC is the substrate peak?

(a) Extremely asymmetric;
(b) Very asymmetric;
(c) Pretty asymmetric;
(d) Just asymmetric; or
(e) Symmetric.

Type the letter of your choice ... ? e

DESCRIPTION OF LAYER PEAK

1. Please type in the Full Width at Half Maximum(FWHM) of the layer peak (in arc seconds) ? 13.89

2. What is the integrated intensity of the layer peak ? 27274

3. How ASYMMETRIC is the layer peak?

(a) Extremely asymmetric;
(b) Very asymmetric;
(c) Pretty asymmetric;
(d) Just asymmetric; or
(e) Symmetric.

Type the letter of your choice ... ? e

4. To what degree is the layer peak WEDGE SHAPED?

(a) There is an extreme degree of wedging;
(b) The peak is very obviously wedge shaped;
(c) The peak is wedge shaped to a degree;
(d) The peak could just about be described as wedge shaped; or
(e) This peak is definitely not wedge shaped.

Type the letter of your choice ... ? e

5. Is the layer peak SPLIT (Y/N) ? n

DESCRIPTION OF THE INTERFERENCE FRINGES

1. Are there INTERFERENCE FRINGES on the rocking curve?

(a) Very large number of fringes;
(b) A significant number of fringes;
(c) Some fringes;
(d) There MAY BE fringes present
BUT I am not quite certain; or
(e) No fringes at all.

Type the letter of your choice ... ? c

2. What is the spacing of the interference fringes (in arc secs) ? 12.5

3. Please describe the VISIBILITY of interference fringes on the layer peak?

(a) Fringes have very high intensity;
(b) Fringes are clearly visible;
(c) Fringes are obvious, but not clear;
(d) Fringes are very unclear; or
(e) Cannot see any fringes at all.

Type the letter of your choice ... ? b

Deductions made by the fuzzy system for a single layer structure of a 1.45 μm thick layer of AlGaAs on a GaAs substrate.

```
THE CONTROL MECHANISMS CHOOSE A SET OF FUZZY RULES AND FIRE THESE RULES
 TO INFER THE STRUCTURAL PARAMETERS

The setfuzz.p program stores default values into structural parameters.

;;; LOADING setfuzz.p

Creating the connection matrix for DKB
Creating the connection matrix for BKT
Creating the connection matrix for NL
Creating the connection matrix for CRT

Scalar multiply each connection matrix by the credibility weight of the
     expert.

Use simple matrix addition to add the connection matrices together.

Loading the ruleset for partition 2 from file

;;; LOADING rules2.p
;;; LOADING rules.p

Firing those rules with the highest weights in the combined connection
     matrix

Fire rules layer by layer and calculate the
 values of the conclusion variables.

The fuzzout.p program calculates the defuzzified output values

;;; LOADING fuzzout.p
;;; LOADING fuzzout.p


The following rules are used in this decision:

[1 6 62 63 69 70 74 79 86 104 107 115]

_____________________________________________________


      DEDUCTIONS MADE BY THE FUZZY SYSTEM

_____________________________________________________

**
_____________________________________________________

   1. GENERAL INFORMATION

The Bragg angle is equal to 33.037 degrees.
The number of layers in this structure is 1.

Press ENTER to continue ...?
_____________________________________________________

   2. CALCULATED VALUES

The experimental mismatch = -304.469 ppm.
The relaxed mismatch = -160.369 ppm.
The spacing of planes = 1.41331
The calculated thickness of the layer = 1.202

Press ENTER to continue ...?
_____________________________________________________

   3. INFERRED CONSEQUENCES

RULE 63 HAS BEEN FIRED
This is the characteristic curve for a single layer structure
This is a curve of two peaks

RULE 6 HAS BEEN FIRED
There is NO evidence of bending of the substrate in this structure
Bending is usually indicated by broadening of the substrate peak

There is NO evidence of grading of the layer in this structure

There is 0.2  evidence of mismatch in this structure.
Please check the calculated result above
 Where the lattice parameters are NOT close, then
 mismatch rather than grading is indicated.
 The opposite is also the case.

There is NO evidence of a change in lattice parameter with depth.

There is 0.6 evidence that the layer is THICK.
 This is indicated by the high intensity of the layer peak.
 It is also indicated by a significant amount of interference fringes.

RULE 1 HAS BEEN FIRED
The quality of this crystal is REASONABLY GOOD.

Press ENTER to continue ...?
_____________________________________________________
```

INPUTS FOR DATA ENTRY WINDOWS IN THE CREATE SCREENS IN RADS

First Window: CONTROL LINE PARAMETERS

Include reference crystal : Using the reference crystal increases the time of the simulation but results in a more accurate graph.

Choose state of polarisation : SAME AS EXPERIMENT

Wavelength : 1.541 Angstroms.

Reflection indices :
H = 0
K = 0
L = 4

Scan range : Start = -500
Finish = 500

The size of the scan step must also be entered.

Second window: SUBSTRATE SELECTION

Substrate material: GaAs

Press ENTER to continue ...?

Third window: REFLECTION GEOMETRY

Selected reflection orientation : Symmetric

Surface normal indices :
H = 0
K = 0
L = 1

Wafer misorientation : There is no obvious evidence of misorientation

---

Fourth window: REFERENCE SELECTION

THIS IS THE SAME AS THE EXPERIMENT

The reference crystal is usually the same material as the substrate.

Press ENTER to continue ...?

Fifth window: OVERLAYER DEFINITION

Number of layers : 1

Sixth window: LAYER CONTROL LINE

Layer Number 1

Layer thickness = 1.45 microns.
Calculated layer thickness = 1.202 microns

Lamellae/layer : Depends on whether composition and strain changes with depth also used when grading occurs ...
As there is NO evidence of grading then use 1 lamella.

Relaxation : Depends on whether MISMATCH is large and whether layer or layers are thick. In this case
Choose relaxation greater than 20 percent.

Total number of blocks : 0

Press ENTER to continue ...?

Seventh window: LAYER DEFINITION

Select layer chemistry : compound

Material for current layer : AlGaAs

Constants defining X composition :
These fields are used when simulating grading ...

There is NO evidence of grading so choose
A = 0; B = 0; C = some constant.

Press ENTER to continue ...?

Eighth window: STRAIN DEFINITION

This window is used for additional strain definition.
Strain is additional to that defined by material composition.

This window is here for when you have something like mechanical damage or ion implantation etc and you can use this window to simulate composition change by means of a strain change ...

```
    For relaxed mismatch:

    -160.369 ppm = At2 + Bt + C where t is height above
                                bottom of layer.

    You can also use experimental mismatch in this equation.

  DEFINE A B and C ACCORDINGLY ...  it is up to the user to
    enter sensible values.

 Press ENTER to continue ...?
```

Question and Answer Session for a Single Layer Theoretical Rocking Curve.

```
PLEASE DESCRIBE THE SIMULATED ROCKING CURVE ...

             A SINGLE LAYER STRUCTURE

             DESCRIPTION OF THE CURVE

1. How many peaks are there in the rocking curve?

     (a) one peak;
     (b) two peaks;
     (c) more than two peaks;
     (d) no peaks whatsoever.

Type the letter of your choice ... ? b

2. Does the layer peak overlap with the substrate peak (Y/N) ? n

2(a). Type in a measure of PEAK SEPARATION between the substrate and
      layer peaks (in arc seconds): ? 198

             DESCRIPTION OF THE SUBSTRATE PEAK

1. Please type in the Full Width at Half Maximum(FWHM) of the substrate
     peak (in arc seconds) ? 6.8

2. What is the integrated intensity of the substrate peak ? 70000

3. How ASYMMETRIC is the substrate peak?

     (a) Extremely asymmetric;
     (b) Very asymmetric;
     (c) Pretty asymmetric;
     (d) Just asymmetric; or
     (e) Symmetric.

Type the letter of your choice ... ? e

             DESCRIPTION OF LAYER PEAK

1. Please type in the Full Width at Half Maximum(FWHM) of the layer peak
     (in arc seconds) ? 15.9

2. What is the integrated intensity of the layer peak ? 20870

3. How ASYMMETRIC is the layer peak?

     (a) Extremely asymmetric;
     (b) Very asymmetric;
     (c) Pretty asymmetric;
     (d) Just asymmetric; or
     (e) Symmetric.

Type the letter of your choice ... ? e

4. To what degree is the layer peak WEDGE SHAPED?

     (a) There is an extreme degree of wedging;
     (b) The peak is very obviously wedge shaped;
     (c) The peak is wedge shaped to a degree;
     (d) The peak could just about be described as wedge shaped; or
     (e) This peak is definitely not wedge shaped.

Type the letter of your choice ... ? e

5. Is the layer peak SPLIT (Y/N)  ? n

             DESCRIPTION OF THE INTERFERENCE FRINGES

1. Are there INTERFERENCE FRINGES on the rocking curve?

     (a) Very large number of fringes;
     (b) A significant number of fringes;
     (c) Some fringes;
     (d) There MAY BE fringes present
           BUT I am not quite certain; or
     (e) No fringes at all.

Type the letter of your choice ... ? b

2. What is the spacing of the interference fringes (in arc secs)  ? 15
```

```
3. Please describe the VISIBILITY of interference fringes on the layer
     peak?

    (a) Fringes have very high intensity;
    (b) Fringes are clearly visible;
    (c) Fringes are obvious, but not clear;
    (d) Fringes are very unclear; or
    (e) Cannot see any fringes at all.

Type the letter of your choice ... ? c

-------------------------------------------------------------------------

Calculating the outcome of the decision ...


CALCULATED OUTCOME OF DECISION : 0.955

Value is between 0 and 1 : threshold of 0.8 is currently set for SUCCESS

The outcome of this expert system decision was SUCCESS.

The following rules were used in this decision :

1 6 62 63 69 70 74 79 86 104 107 115

Press ENTER to continue ...?

-------------------------------------------------------------------------

Do you wish to describe another rocking curve (Y/N) ? n
```

APPENDIX G

Example session of rule generation by inductive learning,

performed by the program 'induce.p'

```
     FUZZY SYSTEM FOR X-RAY ROCKING CURVE ANALYSIS

_____________________________________________________________________

INDUCTIVE LEARNING OF NEW RULES FROM EXAMPLES


Survey substrate only data

  creating membership functions for the consequents

  Check fitness function for CORRECTNESS
  Check fitness function for EFFECTIVENESS
  Check if any of the potential rules already exist
  Check fitness function for SIMPLICITY
  Check fitness function for COVERAGE
  Check fitness function for ROBUSTNESS


Survey single layer data

  creating membership functions for the consequents

  Check fitness function for CORRECTNESS
  Check fitness function for EFFECTIVENESS
  Check if any of the potential rules already exist
  Check fitness function for SIMPLICITY
  Check fitness function for COVERAGE
  Check fitness function for ROBUSTNESS


_____________________________________________________________________

   SINGLE LAYER STRUCTURE :

    NEW FUZZY RULES CREATED BY INDUCTIVE LEARNING

    THESE RULES HAVE BEEN VALIDATED BY FITNESS FUNCTIONS



_____________________________________________________________________

FUZZY RULE:

IF
visibility_of_interference_fringes =  EXTREME

THEN   layer_is_thick  =   VERY

Please state whether this rule should be added to the fuzzy rulebase (Y/N) ?   Y

   Rule checked by fitness functtion for NATURALNESS

_____________________________________________________________________

FUZZY RULE:

IF
substrate_peak_broadening =  NONE
 AND
substrate_asymmetry_in_peak =  NONE
 AND
interference_fringes =  NONE

THEN   layer_is_thin  =   FAIRLY

Please state whether this rule should be added to the fuzzy rulebase (Y/N) ?   Y

   Rule checked by fitness functtion for NATURALNESS

_____________________________________________________________________

FUZZY RULE:

IF
substrate_peak_broadening =  NONE
 AND
substrate_asymmetry_in_peak =  NONE
 AND
interference_fringes =  NONE

THEN   evidence_of_mismatch  =   FAIRLY

Please state whether this rule should be added to the fuzzy rulebase (Y/N) ?   Y

   Rule checked by fitness functtion for NATURALNESS

_____________________________________________________________________
```

```
Survey multiple layers data

   creating membership functions for the consequents

   Check fitness function for CORRECTNESS
   Check fitness function for EFFECTIVENESS
   Check if any of the potential rules already exist
   Check fitness function for SIMPLICITY
   Check fitness function for COVERAGE
   Check fitness function for ROBUSTNESS


Survey MQW and Superlattice data

   creating membership functions for the consequents

   Check fitness function for CORRECTNESS
   Check fitness function for EFFECTIVENESS
   Check if any of the potential rules already exist
   Check fitness function for SIMPLICITY
   Check fitness function for COVERAGE
   Check fitness function for ROBUSTNESS

3 new rules have been generated and added to tthe fuzzy system

:
```

APPENDIX H

Example session for adding rules to the fuzzy system.

2 new rules added using the program 'editrule.p'

```
   ADDING NEW RULES TO THE FUZZY RULEBASE:

Which expert is adding the rules:

   1.  Professor B. K. Tanner
   2.  Professor D. K. Bowen
   3.  Doctor M. Loxley
   4.  Dr. C. R. Thomas
   5.  DUMMY expert

Please type the number of your choice ...? 5

_____________________________________________________________

For which type of structure are you entering new rules:

   1.  Substrate Only Structure
   2.  Single Layer Structure
   3.  Multiple Layers Structure
   4.  MQW Structure or Superlattice

Please type the number of your choice ...? 2

_____________________________________________________________


            ADD NEW RULES TO FUZZY RULEBASE

Single Layer Structure :

How many new rules do you wish to add to the fuzzy rulebase  ? 2

_____________________________________________________________


RULE NUMBER 1


CHOOSE A FUZZY PREMISE FOR THE NEW RULE :

[1]   substrate_peak_broadening
[2]   substrate_asymmetry_in_peak
[3]   layer_asymmetry_in_peak
[4]   layer_wedge_shaped_peak
[5]   interference_fringes
[6]   visibility_of_interference_fringes
[7]   intensity_of_layer_peak
[8]   evidence_of_mismatch
[9]   NO FUZZY PREMISE

Please type the number of the FUZZY PREMISE ...? 2

_____________________________________________________________

 You have chosen the fuzzy premise: substrate_asymmetry_in_peak

_____________________________________________________________


PLEASE CHOOSE A VALUE FOR THIS PREMISE :

[1]   EXTREME
[2]   VERY
[3]   FAIRLY
[4]   SOME
[5]   NONE

Please type the number of the VALUE OF THE PREMISE ...? 1

_____________________________________________________________

You have constructed the following rule SO FAR :

IF substrate_asymmetry_in_peak = EXTREME

_____________________________________________________________


Do you wish to add any further fuzzy premises to the rule (Y/N)? N

_____________________________________________________________
```

```
CHOOSE A BOOLEAN PREMISE FOR THE NEW RULE :

[1]   type_of_structure_is_single_layer
[2]   number_of_peaks_is_one
[3]   number_of_peaks_is_two
[4]   number_of_peaks_is_more_than_two
[5]   number_of_peaks_is_none
[6]   substrate_material_equal_to_layer
[7]   peak_separation_is_low
[8]   layer_split_peak
[9]   layer_integrated_intensity_of_peak_is_zero
[10]  layer_thickness_greater_than_half_micron
[11]  layer_thickness_less_than_5_microns
[12]  peak_splitting_is_zero
[13]  peak_splitting_less_than_three_times_width_of_peak
[14]  relaxed_mismatch_is_high
[15]  peak_splitting_is_high
[16]  spacing_of_interference_fringes_is_low
[17]  NO BOOLEAN PREMISE

Please type the number of the BOOLEAN PREMISE ...? 1

_____________________________________________________________

You have chosen the Boolean premise : type_of_structure_is_single_layer

_____________________________________________________________


PLEASE CHOOSE A VALUE FOR THIS PREMISE :

[1]   TRUE
[2]   FALSE

Please type the number of the value ...? 1

_____________________________________________________________

You have constructed the following rule so far :

IF substrate_asymmetry_in_peak = EXTREME AND
        type_of_structure_is_single_layer = TRUE

_____________________________________________________________


Do you wish to add any further Boolean premises to the rule (Y/N) ? N

_____________________________________________________________



CHOOSE A FUZZY CONSEQUENCE FOR THE NEW RULE :

[1]   crystal_quality
[2]   characteristic_curve
[3]   misorientation_of_substrate
[4]   bending_of_substrate
[5]   grading_of_the_layer
[6]   layer_is_thick
[7]   change_in_lattice_parameter_with_depth
[8]   layer_is_present_in_the_substrate_peak
[9]   layer_is_thin
[10]  evidence_of_mismatch
[11]  peak_is_outside_the_scan_range
[12]  misoriented_or_mismatched_layer
[13]  multiple_layers
[14]  simulation_or_calibration_chart_is_needed
[15]  relaxation
[16]  incorrect_experiment
[17]  thickness_of_layer
[18]  experimental_mismatch

Please type the number of the FUZZY CONSEQUENCE ...? 5

_____________________________________________________________

You have chosen the fuzzy consequence :grading_of_the_layer

_____________________________________________________________


PLEASE CHOOSE A VALUE FOR THIS CONSEQUENCE :

[1]   EXTREME
[2]   VERY
[3]   FAIRLY
[4]   SOME
[5]   NONE

Please type the number of the VALUE OF THE CONSEQUENCE ...? 3

_____________________________________________________________
```

```
You have constructed the following rule :

IF substrate_asymmetry_in_peak = EXTREME AND
        type_of_structure_is_single_layer = TRUE

THEN grading_of_the_layer = FAIRLY

Is this rule OK (Y/N)? Y

RULE NUMBER 2

CHOOSE A FUZZY PREMISE FOR THE NEW RULE :

[1]   substrate_peak_broadening
[2]   substrate_asymmetry_in_peak
[3]   layer_asymmetry_in_peak
[4]   layer_wedge_shaped_peak
[5]   interference_fringes
[6]   visibility_of_interference_fringes
[7]   intensity_of_layer_peak
[8]   evidence_of_mismatch
[9]   NO FUZZY PREMISE

Please type the number of the FUZZY PREMISE ...? 6

You have chosen the fuzzy premise: visibility_of_interference_fringes

PLEASE CHOOSE A VALUE FOR THIS PREMISE :

[1]   EXTREME
[2]   VERY
[3]   FAIRLY
[4]   SOME
[5]   NONE

Please type the number of the VALUE OF THE PREMISE ...? 1

You have constructed the following rule SO FAR :

IF visibility_of_interference_fringes = EXTREME

Do you wish to add any further fuzzy premises to the rule (Y/N)? N

CHOOSE A BOOLEAN PREMISE FOR THE NEW RULE :

[1]   type_of_structure_is_single_layer
[2]   number_of_peaks_is_one
[3]   number_of_peaks_is_two
[4]   number_of_peaks_is_more_than_two
[5]   number_of_peaks_is_none
[6]   substrate_material_equal_to_layer
[7]   peak_separation_is_low
[8]   layer_split_peak
[9]   layer_integrated_intensity_of_peak_is_zero
[10]  layer_thickness_greater_than_half_micron
[11]  layer_thickness_less_than_5_microns
[12]  peak_splitting_is_zero
[13]  peak_splitting_less_than_three_times_width_of_peak
[14]  relaxed_mismatch_is_high
[15]  peak_splitting_is_high
[16]  spacing_of_interference_fringes_is_low
[17]  NO BOOLEAN PREMISE

Please type the number of the BOOLEAN PREMISE ...? 1
```

```
You have chosen the Boolean premise : type_of_structure_is_single_layer

PLEASE CHOOSE A VALUE FOR THIS PREMISE :

[1]   TRUE
[2]   FALSE

Please type the number of the value ...? 1

You have constructed the following rule so far :

IF visibility_of_interference_fringes = EXTREME
        AND type_of_structure_is_single_layer = TRUE

Do you wish to add any further Boolean premises to the rule (Y/N) ? N

CHOOSE A FUZZY CONSEQUENCE FOR THE NEW RULE :

[1]   crystal_quality
[2]   characteristic_curve
[3]   misorientation_of_substrate
[4]   bending_of_substrate
[5]   grading_of_the_layer
[6]   layer_is_thick
[7]   change_in_lattice_parameter_with_depth
[8]   layer_is_present_in_the_substrate_peak
[9]   layer_is_thin
[10]  evidence_of_mismatch
[11]  peak_is_outside_the_scan_range
[12]  misoriented_or_mismatched_layer
[13]  multiple_layers
[14]  simulation_or_calibration_chart_is_needed
[15]  relaxation
[16]  incorrect_experiment
[17]  thickness_of_layer
[18]  experimental_mismatch

Please type the number of the FUZZY CONSEQUENCE ...? 6

You have chosen the fuzzy consequence :layer_is_thick

PLEASE CHOOSE A VALUE FOR THIS CONSEQUENCE :

[1]   EXTREME
[2]   VERY
[3]   FAIRLY
[4]   SOME
[5]   NONE

Please type the number of the VALUE OF THE CONSEQUENCE ...? 2

You have constructed the following rule :

IF visibility_of_interference_fringes = EXTREME
        AND type_of_structure_is_single_layer = TRUE

THEN layer_is_thick = VERY

Is this rule OK (Y/N)? Y

Check if new rules already exist ...
Check if simpler rule already exists ...
  a simpler rule contains a subset of the premises and the
  same consequence.

If either occurs then the new rule is discounted.
```

```
ADDING NEW RULES TO THE FUZZY RULEBASE

      opening data files
      calculating total number of rules when new rules are added
      add the existing rules to the new fuzzy ruleset
      add the new rules to the new fuzzy ruleset

      opening more data files
      increasing the size of  the connection matrices
      updating the connection matrices

      increasing the number of rule histories
      increasing the number of maskings of rules
      increasing the number of fuzzy variables
      increasing the numbers of variable histories

      storing the new values to data files


Do you wish to add rules for a different structure (Y/N)  ? N
:
```

PAGES NOT SCANNED AT THE REQUEST OF THE UNIVERSITY

SEE ORIGINAL COPY OF THE THESIS FOR THIS MATERIAL

REFERENCES

S. Arunkumar and S. Yagneshwar, Knowledge acquisition from examples using maximal representation learning, *Machine Learning: Proceedings of the 7th International Conference* 1-23 June 1990 (eds. B. W. Porter and R. J. Mooney), Morgan Kaufmann, Palo Alto, CA, USA, 2-8 (1990).

Bede Tutorial on X-ray Rocking Curve Analysis, Bede Scientific Instruments Ltd., Lindsey Park, Bowburn, Durham DH6 5PF, England (1992).

H. R. Berenji, Fuzzy logic controllers, in *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (ed. R. R. Yager and L. A. Zadeh), Kluwer Academic Publishers, 69-96 (1992).

F. Bergdano, A. Giordana and L. Saitta, *Machine Learning: An Integrated Framework and its Applications*, Ellis Horwood Limited, England (1991).

D. K. Bowen, M. J. Hill and B. K. Tanner, Characterization of sub-surface structures of double crystal X-ray diffraction, *Material Research Society Symposium Proceedings*, **82**, 447-452 (1987).

E. Cox, Adaptive fuzzy systems, *IEEE Spectrum*, **30**(2), 27-31 (1993).

B. D. Cullity, *Elements of X-ray Diffraction*, Addison-Wesley (1978).

W. C. Daugherity, B. Rathakrishnan and J. Yen, Performance evaluation of a self-tuning controller, *Proceedings of IEEE International Conference on Fuzzy Systems*, San Diego, 389-397 (1992).

P. F. Fewster and C. J. Curling, Composition and lattice-mismatch measurement of thin semiconductor layers by X-ray diffraction, *Journal of Applied Physics*, **62**(10), 4154-4158 (1987).

R. Forsyth (ed.), *Machine Learning*, Chapman and Hall, England (1989).

B. R. Gaines and M. L. G. Shaw, Induction of inference rules for expert systems, *Fuzzy Sets Syst.*, **18**(3), 315-328 (1986).

R. V. Guha and D. B. Lenat, CYC: A midterm report, *Artificial Intelligence (USA)*, **11**(3), 32-59 (1991).

G. Guida and G. Mauri, Evaluating performance and quality of knowledge-based systems: foundation and methodology, *IEEE Transactions on Knowledge and Data Engineering*, 5(2), 204-224 (1993).

M. A. G. Halliwell, M. H. Lyons and M. J. Hill, The interpretation of X-ray rocking curves from II-V semiconductor device structures, *Journal of Crystal Growth*, **68**, 523-631 (1984).

M. Hart, Technology and costs of X-ray diffraction topography, in *Characterization of Crystal Growth Defects by X-Ray Methods* (eds. B. K. Tanner and D. K. Bowen), Plenum Press, New York and London, 474-496 (1980).

Henson, R., *An expert system for X-ray rocking curve analysis using analogical reasoning*, Ph.D. thesis, August 1993, University of Warwick.

C. L. Johnson and S. Feycock, A genetics-based technique for the automated acquisition of expert system rule bases, *Proceedings of IEEE/ACM Conference on Developing and Managing Expert System Programs*, Washington DC, IEEE Computer Society Press, 78-82 (1991).

C. L. Karr and E.J. Gentry, Fuzzy control of pH using genetic algorithms, *IEEE Transactions on Fuzzy Systems*, 1(1), 46-53 (1993).

J. M. Keller and R. Krishnapuram, Fuzzy set methods in computer vision, in *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (ed. R. R. Yager and L. A. Zadeh), Kluwer Academic Publishers, 121-145 (1992).

S. Kocabas, A review of learning, *The Knowledge Engineering Review*, 2, 83-87 (1991).

B. Kosko, Adaptive inference in fuzzy knowledge networks, *Proceedings IEEE 1st Annual International Conference on Neural Networks*, San Diego, II-261 - II-268 (1987).

B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall Inc. (1992).

B. Kosko, Fuzzy systems as universal approximators, *Proceedings of IEEE International Conference on Fuzzy Systems*, San Diego, 1153-1162 (1992).

B. Kosko, *Fuzzy Thinking*, Hyperion (1993).

B. Kosko and S. Isaka, Fuzzy logic, *Scientific American*, July (1993).

C. C. Lee, A self-learning rule-based controller employing approximate reasoning and neural net concepts, *International Journal of Intelligent Systems*, **6**, 71-93 (1991).

G. F. Luger and W. A. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, Benjamin-Cummings (1989).

M. L. Maher, Machine learning in design expert systems, *Proceedings of World Congress on Expert Systems*, Orlando, Florida, U.S.A, Pergamon Press, 728-736 (1991).

E.H. Mamdani and S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, *International Journal of Man-Machine Studies*, 7(1), 1-13 (1975).

M. Minsky, A framework for representing knowledge, in *Readings in Knowledge Representation* (ed. R. J. Brachman and H. J. Levesque), Morgan Kaufmann, 1985, 245-262 (1981).

Y. Nakatani, M. Tsukiyama and T. Fukuda, Case-based reasoning and decision aids for engineering design, *Proceedings of World Congress on Expert Systems*, Orlando, Florida, U.S.A., Pergamon Press, 369-376 (1991).

H. T. Nguyen, V. Kreinovich and D. Tolbert, On robustness of fuzzy logics, *Proceedings of Second IEEE International Conference on Fuzzy Systems*, San Francisco, 512-515 (1993).

D. Partridge, *Artificial Intelligence in the Future of Software Engineering*, Ellis Horwood Limited, England (1986).

D. Partridge and Y. Wilks (editors), *The Foundations of Artificial Intelligence*, Cambridge University Press, England (1990).

D. Partridge, *Engineering Artificial Intelligence Software*, Intellect Books, England, (1992).

T. Partridge, *Knowledge Representation Techniques for Expert Systems*, M.Sc thesis, September 1989, University of Warwick.

T. Partridge and T. Tjahjadi, Combining the heuristics of several experts in an expert system, *Proceedings of the IASTED International Conference on Artificial Intelligence and Neural Networks*, Zurich, Switzerland, 22-25 (1991).

T. Partridge and T. Tjahjadi, A self-adaptive fuzzy system that improves its performance over time, refereed and presented at *IEEE Second International Conference on Fuzzy Systems*, San Francisco, 28 March - 1 April, (1993).

T. Partridge and T. Tjahjadi, A self-evaluating expert system for X-ray rocking curve analysis, *International Journal of Intelligent Systems*, 9, 493-517 (1994).

T. Partridge and T. Tjahjadi, Two self-adaptive techniques for fuzzy systems in changing application domains, submitted to *International Journal of Intelligent Systems* in March 1994.

POPLOG User Guide, Issue 1.3, University of Sussex and Integral Solutions Limited, England (1991).

R. Rada, Gradualness facilitates knowledge refinement, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI**-7(5), 523-530 (1985).

R. Rada, Computers and gradualness: The selfish meme, *Machine Learning*, 5(3), 246-254 (1991).

RADS Operation Manual, Version 1.32, Bede Scientific Instruments Ltd., Lindsey Park, Bowburn, Durham DH6 5PF, England (1992).

R. Reiter, A logic for default reasoning, *Readings in Nonmonotonic Logic* (ed. M. L. Ginsberg), Morgan Kaufmann Publishers Inc., Los Altos, CA, 1987, 68-93 (1980).

E. Sanchez, Fuzzy logic knowledge systems and artificial neural networks in medicine and biology, in *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (ed. R. R. Yager and L. A. Zadeh), Kluwer Academic Publishers, 235-251 (1992).

P. Smyth and R. M. Goodman, An information theoretic approach to rule induction from databases, *IEEE Transactions on Knowledge and Data Engineering*, 4(4), 301-316 (1992).

B. K. Tanner, X-ray scattering from multiple-layer structures forming Bragg-case interferometers, *J. Phys. D: Appl. Phys.*, 26, A151-A155 (1993).

M. Valtorta, Automating rule strengths in expert systems, *Proceedings of European Conference on Artificial Intelligence*, 369-371 (1988).

L. X. Wang and J. M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Transactions on Systems, Man and Cybernetics*, 22(6), 1414-1427 (1992).

R. R. Yager, Expert systems using fuzzy logic, in *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (ed. R.R. Yager and L. A. Zadeh), Kluwer Academic Publishers, 27-44 (1992).

L. A. Zadeh, Commonsense and fuzzy logic, in *The Knowledge Frontier* (ed. C. Cercone and G. McCalla), Springer-Verlag, New York, 103-136 (1987).

L. A. Zadeh, Knowledge representation in fuzzy logic, in *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (ed. R.R. Yager and L. A. Zadeh), Kluwer Academic Publishers, 1-25 (1992).

L. A. Zadeh, The calculus of fuzzy if/then rules, *AI Expert (USA)*, March 1992, 22-27 (1992).

H. J. Zimmermann, *Fuzzy set theory - and its applications*, Kluwer Academic Publishers (1991).